TeamViewer

1E SDK documentation

1E SDK documentation provides the latest information for developing instructions. The 1E SDK is not specific to any single version of 1E, but instead pages will tell you the versions supported by a feature.

Table of contents Example

1 Writing instructions 27

1.1 Getting started with TIMS 27

1.1.1 About TIMS 27

1.1.1.1 TIMS and the Tachyon client 27

1.1.2 Instructions and product packs 28

1.1.3 Installing TIMS 29

1.1.3.1 About TIMS 29

1.1.3.2 Interactive installation 29

1.1.4 Starting TIMS 33

1.1.5 Starting TIMS in SYSTEM context 33

1.1.6 The TIMS user-interface 34

1.1.7 Quick verification that TIMS is working 34

1.1.8 The TIMS menu 35

1.1.8.1 The File menu 35

1.1.8.1.1 Opening and Saving instructions 35

1.1.8.1.1.1 Save warnings 35

1.1.8.1.1.2 Save errors 36

1.1.8.1.2 Importing and exporting scripts 36

1.1.8.2 The Edit menu 36

1.1.8.3 The Tools menu 36

1.1.8.3.1 Agent properties 36

1.1.8.3.2 Loaded modules 37

1.1.8.3.3 Preferences 38

1.1.8.4 The Help menu 39

1.2 Using TIMS 39

1.2.1 Installing and starting TIMS 39

1.2.2 Creating an Instruction XML file 39

1.2.2.1 Validating the file 40

1.2.2.1.1 Mandatory Settings 40

1.2.2.1.1.1 Description 41

1.2.2.1.1.2 Name 41

1.2.2.1.1.3 ReadablePayload 41

1.2.2.1.1.4 Version 41

1.2.2.2 Adding a schema 42
1.2.2.2.1 Alter the column name to a valid name 44
1.2.2.2.2 The 'render as' schema item property 45
1.2.2.3 Adding a task group 45
1.2.2.4 Saving the instruction XML file 46
1.2.2.5 Examining the XML file 46
1.2.3 Adding resources 47
1.2.3.1 Managing embedded resources 48
1.2.3.2 Insert Snippet to Download 48
1.2.3.3 Resource downloading in the TIMS environment 48
1.2.3.4 Edit 49
1.2.3.4.1 Update this resource from a file 49
1.2.3.4.2 Save this resource to a file 49
1.2.3.5 Remove 49
1.2.4 Instruction Parameters 49
1.2.4.1 Adding multiple parameters 53
1.2.4.2 Managing parameters 53
1.2.4.2.1 Insert substitution pattern 53
1.2.4.2.2 Edit... 53
1.2.4.2.3 Remove 53
1.2.4.3 Parameter properties 54
1.2.5 Aggregation 54
1.2.5.1 Configure Aggregation 55
1.2.5.1.1 Enable aggregation and select a 'group by' column 56
1.2.5.1.2 Choose an aggregation operation and a target column 57
1.2.5.2 Computing multiple aggregated results 59
1.2.5.3 Enabling and disabling aggregation 60
1.2.6 Workflow 60
1.2.7 Digital signing 61
1.2.8 Templates 62
1.3 Running instructions for the first time 63
1.3.1 Prefixes, code signing certificates and licensing 63
1.3.1.1 Registering a prefix and code signing certificate and updating the Tachyon license 64
1.3.1.2 Prerequisites 64
1.3.1.3 Assumptions 64

1.3.2 Steps to register a prefix and code signing certificate to obtain a Tachyon license 65
1.3.2.1 Decide your Instruction Name Prefix 65
1.3.2.2 Obtain a code signing certificate 65
1.3.2.3 Import the certificate to the local computer store 66
1.3.2.4 Copy the certificate to the Trusted Publishers store on the Server 67
1.3.2.5 Install TIMS onto the Server 67
1.3.2.6 Edit and Sign the HelloWorldTest instruction 67
1.3.2.7 Import the HelloWorldTest instruction into 69
1.3.2.8 Get the Thumbprint of the code signing certificate and send the Thumbprint and Prefix details to 1E 69
1.3.2.9 Confirm your Server License has been updated 69
1.3.2.10 Confirm the HelloWorldTest instruction is licensed 70
1.3.2.11 Delete the HelloWorldTest instruction 70
1.4 Instruction Definition Reference 71
1.4.1 Terminology 71
1.4.2 Instruction Definition format 72
1.4.3 Digital Signature 75
1.4.4 Sample Instruction Definition XML 75
1.4.5 Instruction Definition XML elements 78
1.4.5.1 InstructionDefinition 79
1.4.5.2 SchemaJson 89
1.4.5.3 Supported Data Types 90
1.4.5.3.1 RenderAs 91
1.4.5.4 AggregationJson 91
1.4.5.4.1 Supported Aggregation Operations 94
1.4.5.5 ParameterJson 94
1.4.5.6 Workflow 98
1.4.5.7 ResponseTemplateConfiguration 98
1.4.5.8 Resources 99
1.4.5.8.1 Resource 99
1.4.5.8.1.1 Content 99
1.4.5.9 TaskGroups 100
1.4.5.9.1 TaskGroup 100
1.4.6 XSD schema 101
1.5 SCALE User Guide 103

1.5.1 Quick Start 103
1.5.2 Key Concepts 105
1.5.2.1 Modules and Methods 105
1.5.2.2 Method Parameters 105
1.5.2.3 Method Output 105
1.5.2.4 SQL 106
1.5.3 Basic syntax 106
1.5.3.1 Simple method execution 106
1.5.3.2 Strings and Escaping 108
1.5.3.3 Passing method parameters 108
1.5.3.4 @tables 109
1.5.3.5 SQL queries 110
1.5.3.6 Reserved keywords 111
1.5.3.7 Comments 111
1.5.4 From Agent Language to Tachyon Instruction 112
1.5.4.1 Writing and testing Instructions 112
1.5.4.2 Instructions and Instruction Sets 112
1.5.4.3 Instruction Parameters 113
1.5.4.4 Handling multiple values in a single string instruction parameter 114
1.5.4.5 Output content and column names 114
1.5.4.6 Aggregating data 115
1.5.5 SELECTing Data 115
1.5.5.1 Limiting Data 115
1.5.5.2 JOINing and GROUPing 116
1.5.5.3 SELECTing "out of thin air" 116
1.5.5.4 More complex examples 116
1.5.5.4.1 Get the top 5 processes with the most number of active TCP connections 116
1.5.5.4.2 String concatenation 117
1.5.5.4.3 Using CASE to translate values 117
1.5.5.4.4 Sub-SELECTs 117
1.5.5.4.5 RECURSIVE CTEs 118
1.5.6 Instruction flow control 118
1.5.6.1 Inconsistent branching results 118
1.5.6.2 Using "EVALUATE" to break on no output 119
1.5.6.3 Looping with FOREACH 120

1.5.6.4 Conditional branching using IF/ELSE/ENDIF 121
1.5.6.4.1 Simple branching 121
1.5.6.4.2 Nesting 122
1.5.6.4.3 IF Conditions 123
1.5.6.4.4 Negating an IF 123
1.5.6.4.5 Capturing the result of an IF statement 124
1.5.6.4.6 Different data from different branches 125
1.5.6.4.7 Do you really need to use an IF? 125
1.5.7 Advanced concepts 126
1.5.7.1 Passing @tables as method parameters 126
1.5.7.2 Agent SQL functions 127
1.5.7.3 Agent built-in functions 129
1.5.7.4 Global Settings variables 129
1.5.7.5 Environment variables 130
1.5.7.6 Querying historic data 130
1.5.8 Instruction termination 131
1.5.8.1 Exit Codes 131
1.5.8.2 Early exit 135
1.5.8.2.1 RETURN 135
1.5.8.2.2 NOCONTENT 136
1.5.8.2.3 ERROR 136
1.5.8.2.4 NOTIMPLEMENTED 137
1.5.9 Datetime handling 137
1.5.9.1 Datetime Functions 137
1.5.9.2 Formats 138
1.5.9.2.1 Converting from Unix Epoch timestamp into readable formats 138
1.5.9.2.2 Converting from readable format into Unix Epoch timestamp 139
1.5.9.2.3 Making data lists 139
1.5.9.3 When is now? 139
1.5.10 Client Activity Record 140
1.5.10.1 What is Client Activity Record? 140
1.5.10.1.1 Capture sources 141
1.5.10.2 How do I retrieve the data from Tachyon client devices? 151
1.5.10.3 How is the data managed? 154
1.5.10.4 Client Activity Record schema 154

1.5.10.4.1 Timestamps 156
1.5.10.4.2 ARP cache entries 156
1.5.10.4.3 Boot performance 157
1.5.10.4.4 Device interaction 161
1.5.10.4.5 Device performance 166
1.5.10.4.6 Device resource demand 177
1.5.10.4.7 DNS resolutions 181
1.5.10.4.8 Operating System performance 182
1.5.10.4.9 Performance event 190
1.5.10.4.10 Process executions 194
1.5.10.4.11 Process stabilizations 197
1.5.10.4.12 Process usage 200
1.5.10.4.13 Sensitive processes 206
1.5.10.4.14 Software installations 208
1.5.10.4.15 Software interaction 210
1.5.10.4.16 Software performance 217
1.5.10.4.17 TCP outbound connections 224
1.5.10.4.18 User usage 227
1.5.11 Tags 231
1.5.11.1 Tag Types 231
1.5.11.2 Tag Storage 232
1.5.11.3 Tag Methods 233
1.5.11.4 Defining Device Tags 233
1.5.12 User Defined Persistent Storage Tables 234
1.5.12.1 Storage Methods 235
1.5.12.2 Examples 235
1.6 Functions and Methods Reference 236
1.6.1 Tachyon Agent Methods 244
1.6.1.1 Agent.CheckForUpdates 255
1.6.1.2 Agent.Echo 256
1.6.1.3 Agent.GetConnectionInformation 257
1.6.1.4 Agent.GetDiagnostics 259
1.6.1.5 Agent.Sleep 261
1.6.1.6 Agent.GetExtensibility 261
1.6.1.7 Agent.GetSummary 262

1.6.1.8 Agent.GetInstructionStatistics 264
1.6.1.9 Agent.Stagger 265
1.6.1.10 Agent.Log 266
1.6.1.11 Agent.GetKeyFiles 267
1.6.1.12 ContentDistribution.Stage 270
1.6.1.13 ContentDistribution.StageAndExecute 271
1.6.1.14 Device.GetCriticality 273
1.6.1.15 Device.GetInboundMappedDriveHistory 274
1.6.1.16 Device.GetDisks 275
1.6.1.17 Device.SetLocation 276
1.6.1.18 Device.GetOutboundPrinters 277
1.6.1.19 Device.GetSummary 278
1.6.1.20 Device.GetLocation 280
1.6.1.21 Device.SetCriticality 280
1.6.1.22 Device.GetOutboundMappedDriveHistory 281
1.6.1.23 Device.GetInboundPrinters 282
1.6.1.24 Device.GetProcessors 283
1.6.1.25 Discovery.LookupNetBiosNames 284
1.6.1.26 Discovery.SendArpRequest 285
1.6.1.27 Discovery.ScanPortsOnDevices 286
1.6.1.28 FileContent.FindAndDelete 289
1.6.1.29 FileContent.GetTorrentDescribedFiles (deprecated) 291
1.6.1.30 FileContent.WriteText 291
1.6.1.31 FileContent.FindAndReplace 293
1.6.1.32 FileContent.FindAndAppend 294
1.6.1.33 FileContent.AppendText 295
1.6.1.34 FileContent.GetTorrentInfo (deprecated) 297
1.6.1.35 FileContent.GetTorrentAnnounceUris (deprecated) 298
1.6.1.36 FileSystem.CreateTempFile 299
1.6.1.37 FileSystem.DeleteFileByName 300
1.6.1.38 FileSystem.GetFilePermissions 301
1.6.1.39 FileSystem.GetFileByLine 305
1.6.1.40 FileSystem.GetDirectoryHash 307
1.6.1.41 FileSystem.GetDigitalSignature 308
1.6.1.42 FileSystem.GetFile 310

1.6.1.43 FileSystem.GetFilesInFolder (deprecated) 311
1.6.1.44 FileSystem.GetHash (deprecated) 313
1.6.1.45 FileSystem.GetFileHash 314
1.6.1.46 FileSystem.FindFileBySizeAndHash 315
1.6.1.47 FileSystem.GetFilesInDirectory 317
1.6.1.48 FileSystem.GetVersionInfo 319
1.6.1.49 FileSystem.PathExists 320
1.6.1.50 FileSystem.FindDirectoryByName 321
1.6.1.51 FileSystem.FindFileByName 322
1.6.1.52 FileSystem.GetFileDetails 327
1.6.1.53 Http.Post 333
1.6.1.54 Interaction.GetActiveResponses 337
1.6.1.55 Interaction.GetInteractionState 339
1.6.1.56 Interaction.GetResponseHistory 340
1.6.1.57 Interaction.GetSessions 342
1.6.1.58 Interaction.GetWindows 342
1.6.1.59 Interaction.InvalidateResponse 343
1.6.1.60 Interaction.SetDoNotDisturb 344
1.6.1.61 Interaction.ShowAnnouncement 345
1.6.1.62 Interaction.ShowNotification 348
1.6.1.63 Interaction.ShowQuestion 349
1.6.1.64 Interaction.ShowSurvey 351
1.6.1.65 Inventory.EvaluateItem 357
1.6.1.66 Inventory.GetLocalReplica 358
1.6.1.67 Inventory.GetReplicationAudit 358
1.6.1.68 Inventory.GetReplicationState 359
1.6.1.69 Inventory.GetTables 360
1.6.1.70 NativeServices.GetWmiNames 360
1.6.1.71 NativeServices.RegistryDeleteKey 362
1.6.1.72 NativeServices.RegistryDeleteUserKey 363
1.6.1.73 NativeServices.RegistryDeleteUserValues 364
1.6.1.74 NativeServices.RegistryDeleteValue 365
1.6.1.75 NativeServices.RegistryEnumerateKeys 366
1.6.1.76 NativeServices.RegistryEnumerateUserKeys 367
1.6.1.77 NativeServices.RegistryEnumerateUserValues 368

1.6.1.78 NativeServices.RegistryEnumerateValues 370
1.6.1.79 NativeServices.RegistryGetUserValues 371
1.6.1.80 NativeServices.RegistryGetValue 373
1.6.1.81 NativeServices.RegistryKeyExists 374
1.6.1.82 NativeServices.RegistrySetUserValues 375
1.6.1.83 NativeServices.RegistrySetValue 376
1.6.1.84 NativeServices.RegistryUserKeyExists 378
1.6.1.85 NativeServices.RegistryUserValueExists 378
1.6.1.86 NativeServices.RunCommand 379
1.6.1.87 NativeServices.RunWmiInstanceMethod 385
1.6.1.88 NativeServices.RunWmiQuery 387
1.6.1.89 NativeServices.RunWmiStaticMethod 389
1.6.1.90 NativeServices.RegistryValueExists 391
1.6.1.91 Network.GetArpEntries 392
1.6.1.92 Network.GetConnections 393
1.6.1.93 Network.GetDnsEntriesByAddress 394
1.6.1.94 Network.GetIpAddress (deprecated) 394
1.6.1.95 Network.GetIpAddresses 395
1.6.1.96 Network.GetListeningEndpoints 397
1.6.1.97 Network.GetWiFiNetworks 398
1.6.1.98 Network.Wakeup 399
1.6.1.99 Nomad.CachedPkgStatus 400
1.6.1.100 Nomad.CancelJob 401
1.6.1.101 Nomad.FindBestPkgCache 401
1.6.1.102 Nomad.GetCachedPkgDetails 402
1.6.1.103 Nomad.GetCachePath 403
1.6.1.104 Nomad.GetCmContentLocations 404
1.6.1.105 Nomad.GetVariable 405
1.6.1.106 Nomad.JobsStatus 406
1.6.1.107 Nomad.Pause 406
1.6.1.108 Nomad.PauseJobQueue 407
1.6.1.109 Nomad.Resume 408
1.6.1.110 Nomad.ResumeJobQueue 409
1.6.1.111 Nomad.RunPkgStatusRequest 410
1.6.1.112 Nomad.RunSsdQuery 411

1.6.1.113 Nomad.Status 412
1.6.1.114 Nomad.Stop 413
1.6.1.115 OperatingSystem.ControlService 413
1.6.1.116 OperatingSystem.GetInstalledUpdates 415
1.6.1.117 OperatingSystem.GetProcesses 416
1.6.1.118 OperatingSystem.GetServiceInfo 417
1.6.1.119 OperatingSystem.GetSummary 420
1.6.1.120 OperatingSystem.GetUpdatesPendingReboot 421
1.6.1.121 OperatingSystem.GetUpdateSummary 422
1.6.1.122 OperatingSystem.KillProcess 423
1.6.1.123 OperatingSystem.KillProcesses 423
1.6.1.124 Patch.Deploy 424
1.6.1.125 Patch.GenerateFullInventoryEvents 429
1.6.1.126 Patch.GetMetadata 430
1.6.1.127 Patch.GetStatus 432
1.6.1.128 Patch.List 435
1.6.1.129 Patch.Restart 443
1.6.1.130 Patch.TestConnection 444
1.6.1.131 Policy.ForceStatusReport 447
1.6.1.132 Policy.GetEventSubscriptions 448
1.6.1.133 Policy.GetRuleEvaluationHistory 448
1.6.1.134 Policy.GetRules 450
1.6.1.135 Policy.GetStatistics 452
1.6.1.136 Policy.GetSummary 454
1.6.1.137 Policy.Invalidate 455
1.6.1.138 Scripting.RunText 456
1.6.1.139 Scripting.Run 462
1.6.1.140 Security.CheckSimpleIoc 469
1.6.1.141 Security.ModifyQuarantineWhitelist 470
1.6.1.142 Security.DeleteCertificates 471
1.6.1.143 Security.GetCertificates 473
1.6.1.144 Security.GetQuarantineStatus 475
1.6.1.145 Security.QuarantineDevice 476
1.6.1.146 Security.UnquarantineDevice 478
1.6.1.147 Shopping.LaunchWsaWizard 479

1.6.1.148 Shopping.ShowNotification 481
1.6.1.149 Software.GetInstallations 483
1.6.1.150 Software.GetUserInstallations 485
1.6.1.151 Software.Uninstall 486
1.6.1.152 Storage.Check 487
1.6.1.153 Storage.List 488
1.6.1.154 Storage.Delete 489
1.6.1.155 Storage.Get 490
1.6.1.156 Storage.GetRemote 491
1.6.1.157 Storage.Set 492
1.6.1.158 Tagging.Check 494
1.6.1.159 Tagging.Clear 495
1.6.1.160 Tagging.Count 496
1.6.1.161 Tagging.Delete 497
1.6.1.162 Tagging.Get 498
1.6.1.163 Tagging.GetAll 499
1.6.1.164 Tagging.Set 501
1.6.1.165 Users.FindLocalGroupMember 502
1.6.1.166 Users.GetInboundConnectionHistory 502
1.6.1.167 Users.GetLocalGroupMembers 503
1.6.1.168 Users.GetLoggedOnUsers 504
1.6.1.169 Users.GetOutboundConnectionHistory 505
1.6.1.170 Users.GetPrimaryUser 506
1.6.1.171 Utilities.GenerateSequence 507
1.6.1.172 Utilities.JsonFromTable 508
1.6.1.173 Utilities.SplitLines 509
1.6.1.174 Utilities.TableFromJson 510
1.6.2 Tachyon Agent Built-in Functions 511
1.6.2.1 HttpGetFile 511
1.6.2.2 Raise 516
1.6.2.3 IsUndefined 518
1.6.3 Tachyon Agent Built-in Keywords 519
1.6.3.1 ERROR 519
1.6.3.2 IF ELSE ENDIF 520
1.6.3.3 RETURN 521

1.6.3.4 EVALUATE 522
1.6.3.5 FOREACH IN DO DONE 522
1.6.3.6 NOCONTENT 523
1.6.3.7 NOTIMPLEMENTED 524
1.6.3.8 true false 524
1.6.4 Tachyon Agent SQL Functions 525
1.6.4.1 BASE64DECODE 525
1.6.4.2 BASE64ENCODE 525
1.6.4.3 BASE64URLENCODE 526
1.6.4.4 BASENAMEFROMPATH 526
1.6.4.5 COMBINEPATHS 527
1.6.4.6 COMPAREVERSIONS 527
1.6.4.7 DATETRUNC 528
1.6.4.8 DIRECTORYFROMPATH 530
1.6.4.9 EPOCHTOJSON 530
1.6.4.10 EXPAND 531
1.6.4.11 HASHSTRING 532
1.6.4.12 IPINRANGE 532
1.6.4.13 IPV4STRTOINT 534
1.6.4.14 NEWID 535
1.6.4.15 REGEXP 535
1.6.4.16 REGEXPCAPTURE 536
1.6.4.17 REGEXPREPLACE 537
1.7 Guaranteed State Triggers 538
1.7.1 Triggers 538
1.7.2 Triggers vs Trigger Templates 539
1.7.2.1 Trigger de-bouncing 542
1.7.3 AgentStartup 543
1.7.4 DirectoryChange 543
1.7.5 FileChange 544
1.7.6 Interval 545
1.7.7 ServiceStatusChange 547
1.7.8 WindowsEventLog 548
1.7.9 WindowsRegistryChange 556
1.8 Custom Response Visualizations 557

1.8.1 Everyone likes a pretty graph 558
1.8.2 Response Visualizations 558
1.8.3 Configuration basics 558
1.8.4 To aggregate or not to aggregate 558
1.8.5 2d vs 3d 559
1.8.6 Single series (2d) charts 559
1.8.7 Multi series (3d) charts 560
1.8.8 Showing Multiple Chart types on one page 563
1.8.9 Custom Response Definitions 565
1.8.9.1 Response Visualization Definition 565
1.8.9.2 Name Property 566
1.8.9.3 TemplateConfigurations Property 566
1.8.9.4 PostProcessors Property 568
1.8.10 Built-in Processors Reference 569
1.8.10.1 Function: ProcessSingleSeries 569
1.8.10.2 Example: ProcessSingleSeries 570
1.8.10.3 Function: ProcessMultiSeries 571
1.8.10.4 Example: ProcessMultiSeries 572
1.8.10.5 Function: ProcessDateTimeSingleSeries 572
1.8.10.6 Example: ProcessDateTimeSingleSeries 573
1.8.10.7 Function: ProcessDateTimeMultiSeries 574
1.8.10.8 Example: ProcessDateTimeMultiSeries 575
1.8.11 Custom Processors 575
1.8.11.1 External code framework 576
1.8.11.2 Script code 576
1.8.11.3 Loaded Assemblies 576
1.8.11.4 Using Types 577
1.8.11.5 Sample processor 577
2 Integrating with 1E 578
2.1 Introduction 578
2.1.1 What is the Consumer API? 578
2.1.2 Accessing the Consumer API 579
2.1.2.1 What am I going to need? 579
2.1.3 How do Consumers work? 579
2.1.3.1 Consumers and licensing 580

2.1.4 Using Consumers 580
2.1.4.1 How does software notify Tachyon what Consumer it is? 580
2.1.4.2 Registering a new Consumer 581
2.1.4.3 Consumers vs Principals 581
2.1.4.4 How do Consumers, Principals and Permissions interact 581
2.1.4.5 What happens when I make a call to Consumer API? 582
2.1.5 Excercise in accessing Consumer API 582
2.1.5.1 Hello world example 582
2.1.5.2 Payload format 584
2.1.6 Downloading the Tachyon SDK 584
2.1.6.1 Tachyon .NET Consumer SDK 585
2.1.6.2 Tachyon Offloading SDK 585
2.1.7 API testing tools 585
2.1.7.1 Postman 585
2.1.7.1.1 Turn off SSL certificate validation 585
2.1.7.1.2 Setting the X-Tachyon-Consumer header field 586
2.1.7.1.3 Configuring Credentials 586
2.1.7.1.4 Gotchas 587
2.1.7.2 Fiddler 587
2.1.7.2.1 Enabling Windows Authentication 587
2.1.7.2.2 Setting the X-Tachyon-Consumer header field 588
2.1.7.3 Insomnia 589
2.1.7.3.1 Setting the X-Tachyon-Consumer header field 589
2.1.7.3.2 Configuring Credentials 589
2.2 Tachyon RunInstruction command-line tool 590
2.2.1 Running an Instruction 590
2.2.1.1 RunInstruction parameters 590
2.2.1.2 Instruction parameters 592
2.2.2 Examples 592
2.2.2.1 Simple example 592
2.2.2.2 Example of Instruction parameters 592
2.2.2.3 Example: offloading the results of an instruction 593
2.2.3 Installation and configuration 593
2.2.3.1 Prerequisites 593
2.2.3.2 Registering a consumer 594

2.2.3.2.1 Go to the Tachyon Portal and navigate to Settings→Configuration→Consumers 595
2.2.3.3 Files 595
2.2.3.4 Tool configuration 596
2.2.4 Diagnostics 596
2.3 Consumer API reference 596
2.3.1 Overview 596
2.3.2 Exploring the API using a web browser 597
2.3.2.1 Swagger URL reference for platform 598
2.3.2.2 Tool settings 599
2.3.2.2.1 Authentication Mechanisms - Introduction 599
2.3.2.2.1.1 Types of Authentication Available 600
2.3.2.2.1.1.1 Integrated Windows Authentication 600
2.3.2.2.1.1.1.1 Integrated Windows Authentication - The Basics 600
2.3.2.2.1.1.1.2 Token based authentication and OAuth support - Contacting the API 600
2.3.2.2.1.1.2 Token based authentication and OAuth support 601
2.3.2.2.1.1.2.1 Token based authentication and OAuth support - The basics 601
2.3.2.2.1.1.2.2 Token based authentication and OAuth support - How the system works 601
2.3.2.2.1.1.2.3 Token based authentication and OAuth support - Obtaining a taken 601
2.3.2.2.1.1.2.3.1 Token based authentication and OAuth support - Interactive login 601
2.3.2.2.1.1.2.3.2 Token based authentication and OAuth support - Non-interactive Login 602
2.3.2.2.1.1.2.3.3 Token based authentication and OAuth support - The toolkit 602
2.3.2.2.1.1.2.4 Token based authentication and OAuth support - Handling Tokens 602
2.3.2.2.1.1.2.4.1 Token based authentication and OAuth support - Token types 602
2.3.2.2.1.1.2.4.2 Token based authentication and OAuth support - Token lifecycle 603
2.3.2.2.1.1.2.5 Token based authentication and OAuth support - Contacting the API 603
2.3.3 Error Conditions 603
2.3.4 Consumer API SDK assembly 604
2.3.5 Release Notes 604

2.3.5.1 Api changes between Tachyon 4.0 and 4.1 604
2.3.5.2 Api changes between Tachyon 4.1 and 5.0 604
2.3.6 ApplicableOperations 604
2.3.7 Approvals 606
2.3.8 AuditLogs 614
2.3.9 Authentication 615
2.3.10 ComponentHealthManager 618
2.3.11 Consumers 618
2.3.12 CustomProperties 630
2.3.13 CustomPropertyTypes 636
2.3.14 CustomPropertyValues 640
2.3.15 DataExport 644
2.3.16 Deploy 647
2.3.16.1 Enumerations 666
2.3.16.1.1 Operating System Type 666
2.3.16.1.2 Operating System Architecture 667
2.3.17 Devices 667
2.3.18 EventSource 686
2.3.19 EventSubscriptionAssignments 694
2.3.20 EventSubscriptions 695
2.3.21 Expressions 700
2.3.22 Fragments 701
2.3.23 Information (default) 704
2.3.24 InstructionDefinitions 705
2.3.25 InstructionHierarchies 717
2.3.26 Instructions 719
2.3.27 InstructionSets 734
2.3.28 InstructionStatistics 740
2.3.29 ManagementGroups 744
2.3.30 Notifications 773
2.3.31 Permissions 776
2.3.32 Policies 786
2.3.33 Policy 791
2.3.34 PolicyAssignments 805
2.3.35 PrincipalRoleManagementGroups 806

2.3.36 PrincipalRoles 813
2.3.37 Principals 817
2.3.38 PrincipalSearch 821
2.3.39 ProductPacks 825
2.3.40 PushMessageUsers (removed) 829
2.3.40.1 User status values: 830
2.3.41 RelatedInstructionDefinitions 830
2.3.42 Resources 831
2.3.43 ResponseErrors 833
2.3.44 Responses 835
2.3.45 ResponseTemplates 842
2.3.46 Role 842
2.3.47 Rules 863
2.3.48 ScheduledInstructions 866
2.3.49 SecurableTypes 875
2.3.50 Settings 878
2.3.51 SystemInformation 879
2.3.52 SystemStatistics 886
2.3.53 Tasks 888
2.3.54 TriggerTemplates 889
2.4 Using scope and filter expressions 891
2.4.1 Building an expression 891
2.4.1.1 Anatomy of an expression 891
2.4.1.2 Available expression operators 892
2.4.2 Expression object 892
2.4.2.1 Expression tree object 893
2.4.3 Examples 894
2.4.3.1 Single element expression 894
2.4.3.2 Multi element expression 894
2.4.3.3 Multi level expression 894
2.4.3.4 Negating an expression 895
2.4.4 Defining a filter 896
2.4.4.1 Where are expressions used 896
2.4.4.1.1 Expressions used to define API filters 896
2.4.4.1.2 Expressions used to define scope 896

2.4.4.1.3 Expressions used to define instruction result filters 896
2.4.4.2 Results filter for an instruction 896
2.4.4.2.1 Use case for filters 896
2.4.4.2.1.1 Example use case 897
2.4.4.2.2 How filters work 897
2.4.4.2.2.1 Results filter 897
2.4.4.2.2.1.1 Example 897
2.4.4.2.2.2 Previous results filter 897
2.4.4.2.2.2.1 Simple example 898
2.4.4.2.2.2.2 Complex example 898
2.4.4.2.2.3 Combining filter types 898
2.4.4.2.3 Using results filter 899
2.4.4.2.3.1 Example 899
2.4.4.2.4 Using previous results filter 900
2.4.4.2.4.1 Example 900
2.4.4.2.5 Putting it all together 901
2.4.5 Defining the scope 903
2.4.5.1 Available scope attributes 903
2.4.5.1.1 Scope expression limitations pre v3.2 904
2.4.5.2 Handling tags 904
2.4.5.3 Handling Management Groups 904
2.4.5.4 Handling Criticality 905
2.4.6 Sort Definition 905
2.4.6.1 Building sort expression 905
2.5 How to guides 906
2.5.1 Authenticate against the 1E Platform and obtain a token 906
2.5.1.1 Authentication via IdP's web page 906
2.5.1.2 Authentication using a JWT 906
2.5.1.3 Implementing interactive authentication 907
2.5.1.3.1 How it works 907
2.5.1.3.2 Before we begin 907
2.5.1.3.3 Authentication request 907
2.5.1.3.4 Examining the response to initial authentication request 908
2.5.1.3.5 Authenticating against the IdP 908
2.5.1.3.6 Checking the status of your authentication request 908

2.5.1.3.7 Retrieving the token 909
2.5.1.3.8 What if something goes wrong? 910
2.5.1.3.9 Consumer SDK support 910
2.5.1.3.10 Implementing interactive authentication - C# example 910
2.5.1.3.10.1 Expand for C# example 910
2.5.1.4 Implementing non-interactive (JWT) authentication 916
2.5.1.4.1 How it works 916
2.5.1.4.2 Configuration 917
2.5.1.4.3 Authentication steps 918
2.5.1.4.4 Non-interactive authentication - Principal to certificate mapping 920
2.5.1.4.4.1 About certificate to principal mapping 920
2.5.1.4.4.2 Connecting to the platform 921
2.5.1.4.4.3 Advanced principal mapping options 922
2.5.1.4.4.4 Verifying mappings 923
2.5.1.4.4.5 Using a dedicated identity provider application for external integration 923
2.5.1.4.4.6 Specifying a dedicated identity provider application for external integration 924
2.5.1.4.5 Implementing non-interactive (JWT) authentication - C# example 924
2.5.1.4.5.1 Before starting 925
2.5.1.4.6 Implementing non-interactive (JWT) authentication - Python example 929
2.5.1.4.6.1 Setting up the environment 930
2.5.1.4.7 Implementing non-interactive (JWT) authentication - Typescript example 935
2.5.2 Add and remove Instruction Definitions and organize them into Instruction Sets 937
2.5.2.1 Instruction Definitions 937
2.5.2.1.1 Uploading Instruction Definitions 937
2.5.2.1.2 Deleting Instruction Definitions 939
2.5.2.2 Instruction Sets 941
2.5.2.2.1 Add an Instruction Set 941
2.5.2.2.2 Edit an Instruction Set 944
2.5.2.2.3 Delete an Instruction Set 945
2.5.2.2.4 Get Instruction Sets 945
2.5.2.2.5 Add Instruction Definitions to an Instruction Set 947
2.5.2.2.6 Remove Instruction Definitions from an Instruction Set 952
2.5.2.2.7 Export Instruction Definitions from a set 953
2.5.3 Find the right instruction definition 953

2.5.3.1 What is an instruction definition? 954
2.5.3.1.1 What properties can be used to uniquely identify an instruction definition? 954
2.5.3.1.2 How is an instruction created from an instruction definition 954
2.5.3.2 Searching for instruction definitions 955
2.5.3.2.1 Retrieving all instruction definitions 955
2.5.3.2.2 Using search-as-you-type feature 962
2.5.3.2.3 Searching for instruction definitions that match specific criteria 963
2.5.3.3 Searching for instruction definitions when issuing a follow-up instruction 972
2.5.3.3.1 Retrieving all instruction definitions that can be used to issue a follow-up 972
2.5.3.3.2 Searching for feasible follow-up instruction definitions that match specific criteria 973
2.5.3.4 Retrieving specific instruction definition 974
2.5.3.5 Exporting Instruction Definitions 977
2.5.4 Getting data from Tachyon clients 979
2.5.4.1 Find all computers 979
2.5.4.2 Finding a specific computer or computers 982
2.5.4.2.1 Filtering and sorting with pagination 982
2.5.4.2.2 Finding specific computers 986
2.5.4.2.2.1 Finding a Device by its Fully Qualified Domain Name (FQDN) 986
2.5.4.2.2.2 Finding a Device by its Tachyon GUID 988
2.5.4.2.3 Finding management groups for computers 989
2.5.4.2.3.1 By Fully Qualified Domain Name (FQDN) 989
2.5.4.2.3.2 By Tachyon GUID 990
2.5.4.3 Find computers matching scope 990
2.5.4.4 Find an approximate target for an instruction 993
2.5.5 Handling errors returned by Tachyon 995
2.5.5.1 What does the Consumer API return when an error happens? 995
2.5.5.2 The Error object 995
2.5.5.2.1 Error code 996
2.5.5.2.2 Localization 996
2.5.5.3 Example error 996
2.5.5.4 Errors in Consumer API SDK 996
2.5.5.5 Finding out more about the error 997
2.5.6 Issue an instruction, track its progress and retrieve responses 997
2.5.6.1 Issuing an instruction 997

2.5.6.1.1 Simple instruction 998
2.5.6.1.2 Adding parameters 1002
2.5.6.1.3 Narrowing the scope 1006
2.5.6.1.4 Targeting specific devices 1010
2.5.6.1.5 Following up on an instruction 1014
2.5.6.1.6 Adding results filters 1017
2.5.6.2 Tracking progress 1021
2.5.6.2.1 Checking Instruction status 1023
2.5.6.2.2 Checking instruction statistics 1026
2.5.6.3 Retrieving responses 1029
2.5.6.3.1 Pagination 1030
2.5.6.3.2 How can I tell if I'm "done" with responses? 1030
2.5.6.3.3 How can I tell if my instruction is aggregated or not? 1031
2.5.6.3.4 Raw responses 1031
2.5.6.3.5 Aggregated responses 1033
2.5.6.3.6 Other responses 1035
2.5.6.3.7 Examining a Response object 1039
2.5.6.3.8 Reading responses 1040
2.5.6.4 Switching from high level to detailed view of responses 1042
2.5.6.4.1 Processed responses 1049
2.5.6.5 Cancelling and re-running instructions 1049
2.5.6.5.1 Re-running instructions 1050
2.5.6.6 Pseudo-code examples 1054
2.5.6.6.1 Example 1: Find a machine with specific hardware and perform an action on it. 1054
2.5.6.6.2 Example 2: Retrieving aggregated responses 1056
2.5.6.6.3 Example 3: Retrieving raw responses 1057
2.5.7 Set up in instruction to be executed on a schedule 1059
2.5.7.1 What is a "scheduled instruction" 1060
2.5.7.2 Creating a schedule 1060
2.5.7.2.1 Core of the scheduled instruction 1060
2.5.7.2.2 Scheduling properties 1060
2.5.7.3 Managing schedules 1065
2.5.7.3.1 Retrieving scheduled instructions 1065
2.5.7.3.2 Retrieving past and/or planned instructions 1072
2.5.7.3.3 Finding all schedule activations for a specific instruction definition 1074

2.5.7.3.4 Finding all schedule activations for a specific scheduled instruction 1075
2.5.7.3.5 Finding all schedule activations that contain a marker 1075
2.5.7.3.6 Complex example 1075
2.5.7.4 Lifecycle of a schedule 1076
2.5.7.5 Create a new schedule 1078
2.5.7.5.1 ScheduleActiveStartTime and ScheduleActiveEndTime 1082
2.5.7.5.2 Narrowing the set of devices that should respond to the scheduled instruction 1082
2.5.7.5.3 Other instruction properties 1082
2.5.7.6 Modifying and cancelling a schedule 1082
2.5.7.7 Example schedules 1084
2.5.8 Set up Principals, Roles and Permissions, for Tachyon version 8.0 and later 1086
2.5.8.1 Basics of role based access control (RBAC) 1086
2.5.8.2 RBAC objects in Tachyon 1086
2.5.8.2.1 Principals 1087
2.5.8.2.2 Roles 1087
2.5.8.2.3 Permissions 1087
2.5.8.2.4 Securable Types 1087
2.5.8.2.5 Operations 1087
2.5.8.2.6 Management Group 1087
2.5.8.2.7 Example how to read permissions 1088
2.5.8.3 Changes introduced in Tachyon version 8.0 1088
2.5.8.3.1 Principals, Roles and Management Groups 1088
2.5.8.3.2 Global and Local Permissions 1089
2.5.8.3.3 Role types 1089
2.5.8.3.4 Anatomy of an RBAC assignment 1089
2.5.8.3.5 Delegation 1090
2.5.8.3.5.1 Types of security administrator 1090
2.5.8.3.5.2 Who can assign what? 1090
2.5.8.3.5.3 An example of how it works 1090
2.5.8.4 Retrieving RBAC objects 1091
2.5.8.4.1 Getting Principals 1091
2.5.8.4.2 Getting Roles 1094
2.5.8.4.3 Getting Securable Types 1106
2.5.8.4.4 Getting Applicable Operations 1132
2.5.8.4.5 Getting Permissions 1133

2.5.8.4.5.1 Getting permissions for a Principal 1133
2.5.8.4.5.2 Getting permissions for a Role 1141
2.5.8.4.5.3 Getting permissions for a Securable Type 1144
2.5.8.4.5.4 Helper APIs 1148
2.5.8.4.5.5 Retrieving Active Directory information about a specific account 1154
2.5.8.5 Configuring Tachyon RBAC through the Consumer API 1155
2.5.8.5.1 Adding Principals from Active Directory 1155
2.5.8.5.2 Configuring Roles 1157
2.5.8.5.3 Configuring Assignments 1166
2.5.8.5.4 Configuring Securable Types and Applicable Operations 1187
2.5.9 Set up Principals, Roles and Permissions, up to and including Tachyon version 5.2 1191
2.5.9.1 Basics of role based access control (RBAC) 1191
2.5.9.2 RBAC objects in Tachyon 1191
2.5.9.2.1 Principals 1191
2.5.9.2.2 Roles 1192
2.5.9.2.3 Permissions 1192
2.5.9.2.4 Securable Types 1192
2.5.9.2.5 Operations 1192
2.5.9.2.6 Example how to read permissions 1192
2.5.9.3 Retrieving RBAC objects 1192
2.5.9.3.1 Getting Principals 1192
2.5.9.3.2 Getting Roles 1195
2.5.9.3.3 Getting Securable Types 1207
2.5.9.3.4 Getting Applicable Operations 1222
2.5.9.3.5 Getting Permissions 1223
2.5.9.3.5.1 Getting permissions for a Principal 1223
2.5.9.3.5.2 Getting permissions for a Role 1228
2.5.9.3.5.3 Getting permissions for a Securable Type 1232
2.5.9.3.5.4 Helper APIs 1236
2.5.9.3.5.5 Retrieving Active Directory information about a specific account 1242
2.5.9.4 Configuring Tachyon RBAC through the Consumer API 1243
2.5.9.4.1 1243
2.5.9.4.1.1 Adding Principals from Active Directory 1243
2.5.9.4.2 Configuring Roles 1245
2.5.9.4.3 Configuring Management Groups 1267

2.5.9.4.4 Configuring Securable Types and Applicable Operations 1269
2.5.9.5 Introduction 1273
2.5.9.6 Basics of role based access control (RBAC) 1273
2.5.9.7 RBAC objects in Tachyon 1273
2.5.9.7.1 Principals 1273
2.5.9.7.2 Roles 1274
2.5.9.7.3 Permissions 1274
2.5.9.7.4 Securable Types 1274
2.5.9.7.5 Operations 1274
2.5.9.7.6 Example how to read permissions 1274
2.5.9.8 Retrieving RBAC objects 1274
2.5.9.8.1 Getting Principals 1274
2.5.9.8.2 Getting Roles 1277
2.5.9.8.3 Getting Securable Types 1289
2.5.9.8.4 Getting Applicable Operations 1304
2.5.9.8.5 Getting Permissions 1305
2.5.9.8.5.1 Getting permissions for a Principal 1305
2.5.9.8.5.2 Getting permissions for a Role 1310
2.5.9.8.5.3 Getting permissions for a Securable Type 1314
2.5.9.8.5.4 Helper APIs 1318
2.5.9.8.5.5 Retrieving Active Directory information about a specific account 1324
2.5.9.9 Configuring Tachyon RBAC through the Consumer API 1325
2.5.9.9.1 1325
2.5.9.9.1.1 Adding Principals from Active Directory 1325
2.5.9.9.2 Configuring Roles 1327
2.5.9.9.3 Configuring Management Groups 1349
2.5.9.9.4 Configuring Securable Types and Applicable Operations 1351
2.6 Training Lab Guide - Using the Application Programming Interface 1355
3 .NET Consumer SDK 1355
3.1 Downloading and installing the SDK 1355
3.2 Consumer SDK library 1355
3.2.1 SDK versioning 1356
3.2.2 TachyonConnector 1356
3.2.2.1 TachyonConnector constructors 1356
3.2.3 Code structure 1356

3.2.3.1 APIs 1356
3.2.3.2 Models 1356
3.2.3.3 Available customization interfaces 1358
3.2.3.4 Default implementations 1358
3.2.3.5 Tools 1359
3.2.4 Code examples 1359
3.2.4.1 Example 1: Access Tachyon using default credentials 1359
3.2.4.2 Example 2: Access Tachyon using specific credentials 1359
3.2.4.3 Example 3: Access Tachyon using a custom transport proxy 1360
3.2.4.4 Example 4: A complete example 1360
3.3 Offloading SDK library 1362
3.3.1 Creating an Offloading Target 1362
3.4 SDK release information 1363
3.4.1 .NET Consumer SDK 8.1 release notes – last updated 18-August-2022 1363
3.4.1.1 Online help 1363
3.4.2 Tachyon .NET Consumer SDK release history 1364

Provides the latest information for developing instructions. The is not specific to any single version of, but instead pages will tell you the versions supported by a feature.

Please visit the 1E Community forum (in the 1E Support Portal) to ask questions and provide feedback.

1 Writing instructions

How to write instructions using and DEXCode and TIMS).

The main purpose of is to enable super-fast investigation of your network environment to retrieve sets of devices that you can then run immediate actions on.

> **Note**
>
> This documentation refers to the Tachy DEXCode and TIMS).on Agent, which is an old name for . It provides client modules for , , Shopping/WSA and WakeUp. The Nomad client module has some methods available for use in .

1.1 Getting started with TIMS

Steps to create a simple instruction using Tachyon Instruction Management Studio (TIMS).

1.1.1 About TIMS

TIMS is a graphical user interface for authoring and testing Tachyon instructions, which can then be saved so they can be uploaded and used in your Tachyon system.

The instruction definition XML files that TIMS creates can be uploaded into Tachyon using the Tachyon Portal Setting application so that they can then be used within Tachyon, or combined with other instructions into a Product Pack zip.

TIMS makes it easy for you to:

- Run instructions and verify the results they return from your client devices
- Manage resources (such as files) which are associated with the instruction (question or action) you are developing
- Save an instruction, and resources, as an XML file, which can be loaded into Tachyon and made available as a question or action
- Sign an instruction with a different code signing certificate.

1.1.1.1 TIMS and the Tachyon client

TIMS contains a copy of the **1E Client**, isolated from the local 1E Client installed on your computer. It has only the Tachyon client features enabled, other client modules are disabled. Earlier versions of TIMS contained a copy of **Tachyon Agent**.

Each version of TIMS is bound to its version of 1E Client, which does not have to be the same version as your local client, nor does it have to be the same architecture. This means you can have multiple versions of TIMS

installed, and you can test your instructions against different versions of the client that you may have in your Tachyon system, or are planning to have. However, you can run only one copy of TIMS at any time.

Note

Currently, TIMS is supported only on Windows OS.

For supported OS versions and other requirements please refer to version specific documentation.

The *Tachyon Activity Record feature* is enabled (module.inventory.enabled=true). It has its own database (Engine.Inventory.dat) which does not affect the database of the local client (Inventory.dat). This means you need to leave TIMS running if you want to accumulate data to test with, and each time you start TIMS it will continue adding to the database. If you are testing instructions with older versions of TIMS then take care not to be confused by data captured by later versions.

Note

This documentation refers to the Tachyon Agent, which is an old name for . It provides client modules for , , Shopping/WSA and WakeUp. The Nomad client module has some methods available for use in .

1.1.2 Instructions and product packs

Instructions are developed and signed in TIMS. The name of the instruction and how it is signed governs whether it can be uploaded and run in Tachyon. Once signed, the instruction XML file can be uploaded directly into Tachyon using the Tachyon Portal Settings application.

- **Classic product packs** are zips containing one or more Tachyon instructions. These are uploaded into Tachyon using the Tachyon Portal Settings application.
- **Integrated product packs** are zips containing Guaranteed State policies and may also contain Tachyon instructions. These are uploaded in Tachyon using the **Product Pack Deployment Tool**. The tool can also be used to upload classic product packs but the documentation has not been updated to explain this.

Important

When any policy artifacts are modified, the Policy infrastructure is flagged as "dirty," triggering a prompt to Deploy Policy. This occurs even if the modified artifacts are not in use by any active Policy. If you're confident that none of the modified artifacts are in use, deploying is unnecessary. If unsure, you can safely proceed with the deployment, as it won't affect anything if there are no changes.

Code signing certificates

Instructions do not need to be signed if you only want to develop and test instructions locally in TIMS, or share them with other developers.

Instructions must be signed with an appropriate code signing certificate if you want to import instructions into a Tachyon system and run them. An appropriate code signing certificate is one that has been registered in your Tachyon license to be used with

instructions that have a specific prefix. For more details about code signing certificates, prefixes and licensing, please refer to Running instructions in Tachyon for the first time.

Known issue: The current version of TIMS does not support CNG code signing certificates. You can select a CNG certificate but when you attempt to save an instruction then you will see an error: *Tachyon Instruction Management Studio: Warning. The file <xml file> could not be digitally signed. Invalid provider type specified.*

1.1.3 Installing TIMS

Installing Tachyon Instruction Management Studio (TIMS).

The TIMS installer is downloaded from the 1E Support Portal as a zip containing x86 and x64 MSI installers.

You should install the same or later version of TIMS as the 1E Client you want to develop instructions for. The latest version of TIMS will be suitable because SCALE is backwards-compatible in most cases. Functions and Methods Reference tells you the earliest version in which a SCALE function or method was implemented, and on which OS.

After installing TIMS you can begin development of Tachyon instructions. A good place to start is Writing instructions.

1.1.3.1 About TIMS

Tachyon Instruction Management Studio (TIMS) is a graphical user interface for authoring and testing Tachyon instructions, which can then be saved so they can be uploaded and used in your Tachyon system.

The instruction definition XML files that TIMS creates can be uploaded into Tachyon using the Tachyon Portal Setting application so that they can then be used within Tachyon, or combined with other instructions into a Product Pack zip.

TIMS contains a copy of the **1E Client**, isolated from the local 1E Client installed on your computer. It has only the Tachyon client features enabled, other client modules are disabled. Earlier versions of TIMS contained a copy of **Tachyon Agent**.

Each version of TIMS is bound to its version of 1E Client, which does not have to be the same version as your local client, nor does it have to be the same architecture. This means you can have multiple versions of TIMS installed, and you can test your instructions against different versions of the client that you may have in your Tachyon system, or are planning to have. However, you can run only one copy of TIMS at any time.

Note

Currently, TIMS is supported only on Windows OS.

For supported OS versions and other requirements please refer to version specific documentation.

1.1.3.2 Interactive installation

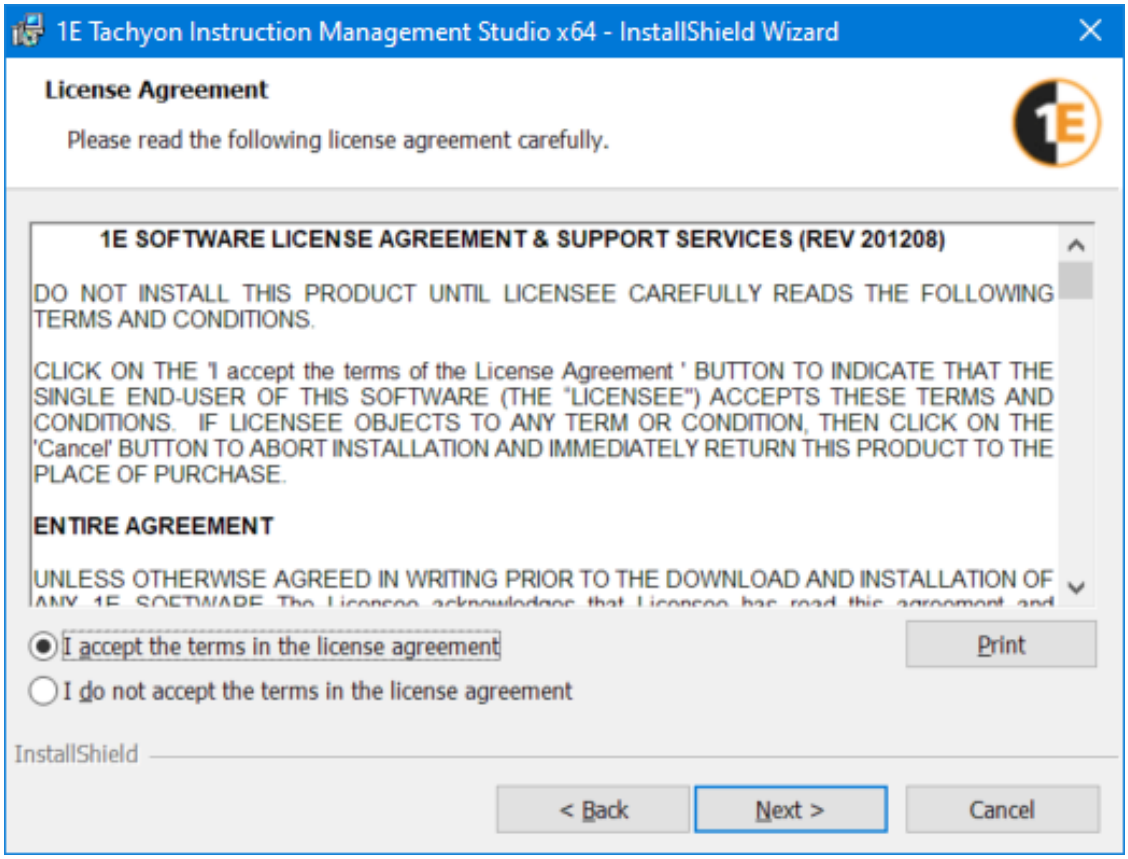
During installation you may see a UAC prompt for 1E Client.

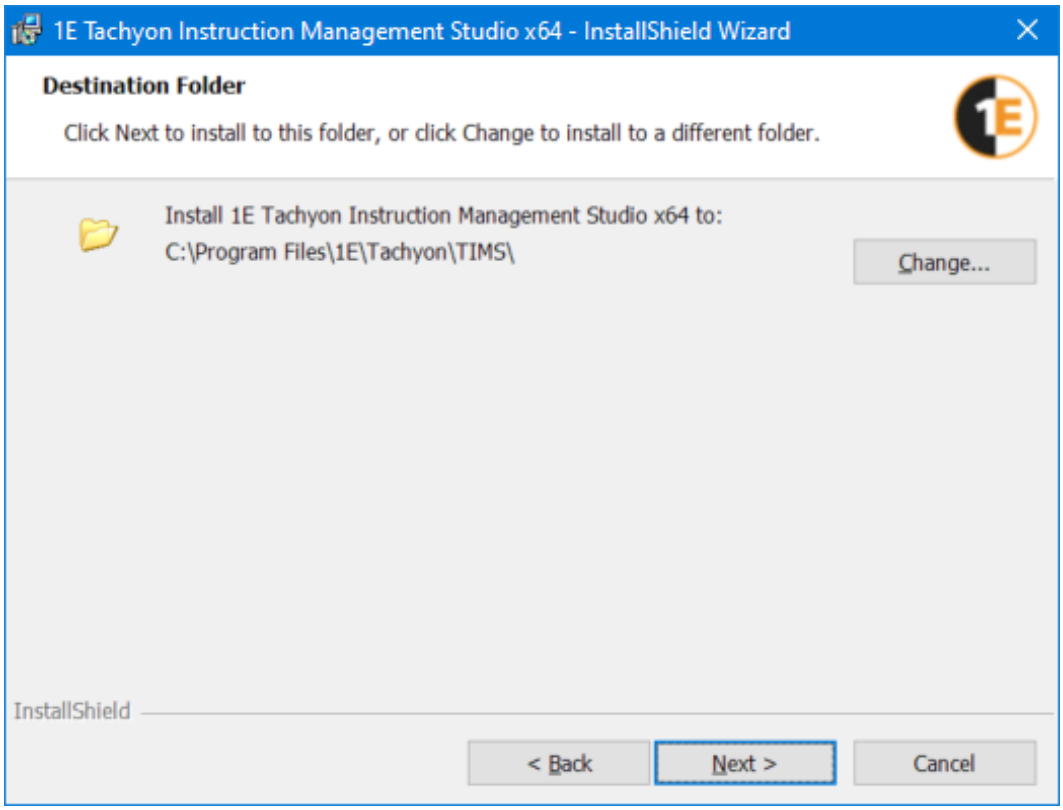
The default installation location is **C:\Program Files\1E\Tachyon\TIMS** - you can safely rename this folder to be version specific - say TIMS_v5.0.0.745_x64.

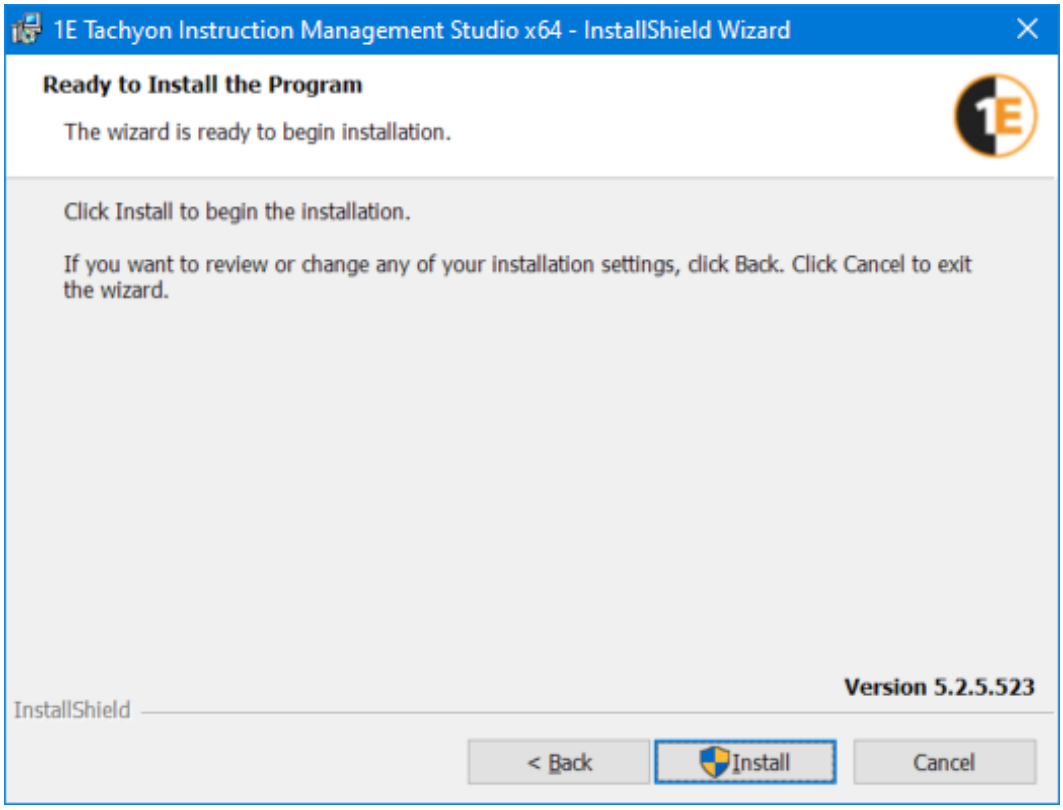
A shortcut named **Tachyon Instruction Management Studio** is added to the public desktop (C:\Users\Public\Desktop) - which runs **Tachyon.InstructionManagementStudio.exe**

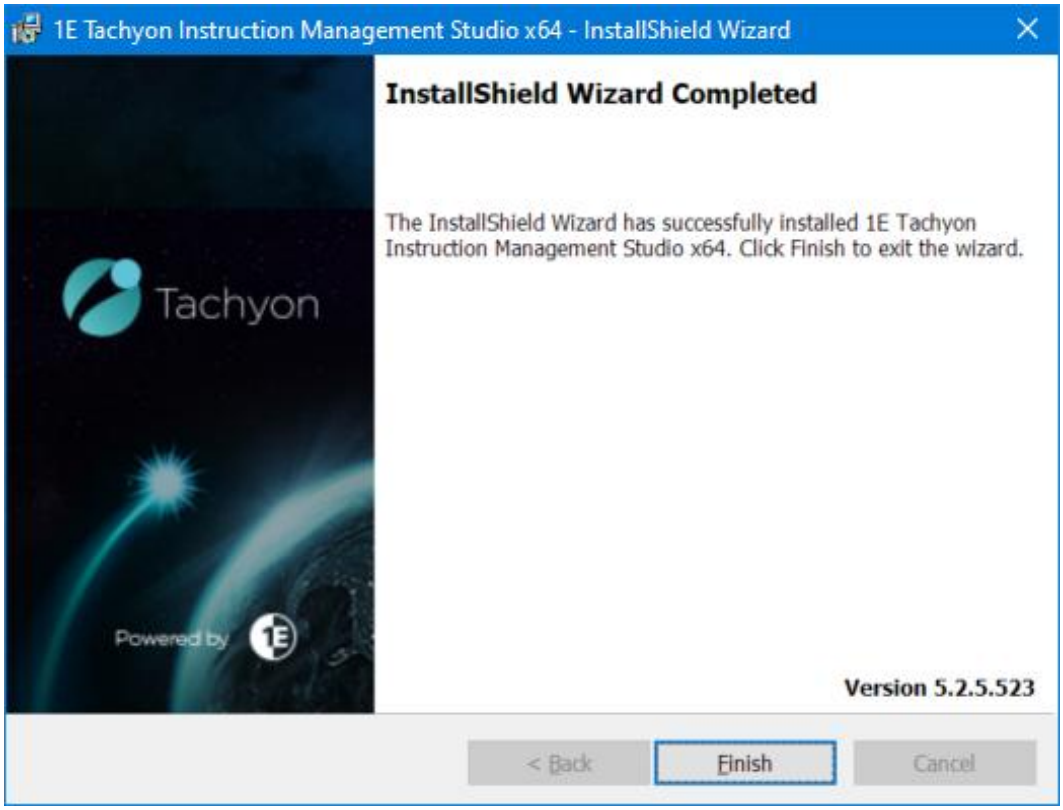
You can only have one version of TIMS installed or running at the same time. However, you can safely make copies of the TIMS installation folder and create your own shortcuts, if you want to keep multiple versions of TIMS on the same computer.




1E Tachyon Instruction Management Studio x64 - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
1E SOFTWARE LICENSE AGREEMENT & SUPPORT SERVICES (REV 201208)
DO NOT INSTALL THIS PRODUCT UNTIL LICENSEE CAREFULLY READS THE FOLLOWING TERMS AND CONDITIONS.
CLICK ON THE 'I accept the terms of the License Agreement ' BUTTON TO INDICATE THAT THE SINGLE END-USER OF THIS SOFTWARE (THE "LICENSEE") ACCEPTS THESE TERMS AND CONDITIONS. IF LICENSEE OBJECTS TO ANY TERM OR CONDITION, THEN CLICK ON THE 'Cancel' BUTTON TO ABORT INSTALLATION AND IMMEDIATELY RETURN THIS PRODUCT TO THE PLACE OF PURCHASE.
ENTIRE AGREEMENT
UNLESS OTHERWISE AGREED IN WRITING PRIOR TO THE DOWNLOAD AND INSTALLATION OF
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Print
InstallShield
< Back
Next >
Cancel


1E Tachyon Instruction Management Studio x64 - InstallShield Wizard
Destination Folder
Click Next to install to this folder, or click Change to install to a different folder.
Install 1E Tachyon Instruction Management Studio x64 to:
C:\Program Files\1E\Tachyon\TIMS\
Change...
InstallShield
< Back
Next >
Cancel


1E Tachyon Instruction Management Studio x64 - InstallShield Wizard
Ready to Install the Program
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
Version 5.2.5.523
InstallShield
< Back
Install
Cancel


1E Tachyon Instruction Management Studio x64 - InstallShield Wizard
Tachyon
Powered by
InstallShield Wizard Completed
The InstallShield Wizard has successfully installed 1E Tachyon Instruction Management Studio x64. Click Finish to exit the wizard.
Version 5.2.5.523
< Back
Finish
Cancel

1.1.4 Starting TIMS

Double-click on the shortcut or executable **Tachyon.InstructionManagementStudio.exe** you may see a UAC prompt from the 1E Client.

1.1.5 Starting TIMS in SYSTEM context

Sometimes you may need to develop and test your instructions in SYSTEM context, in order to replicate how the Tachyon client runs actually runs.

You can do this by starting TIMS using Microsoft SysInternal's PSEXEC utility.

1. Copy PSEXEC utility to a known path, eg. Windows\System32 folder, or to the same folder as TIMS.
2. In the TIMS folder, create a command file called **StartTIMS.cmd** containing the instructions shown below
3. Create a shortcut to the cmd file, and use this to start TIMS using **Run As Administrator**
4. If you are using PSEXEC64 then edit the cmd file accordingly.

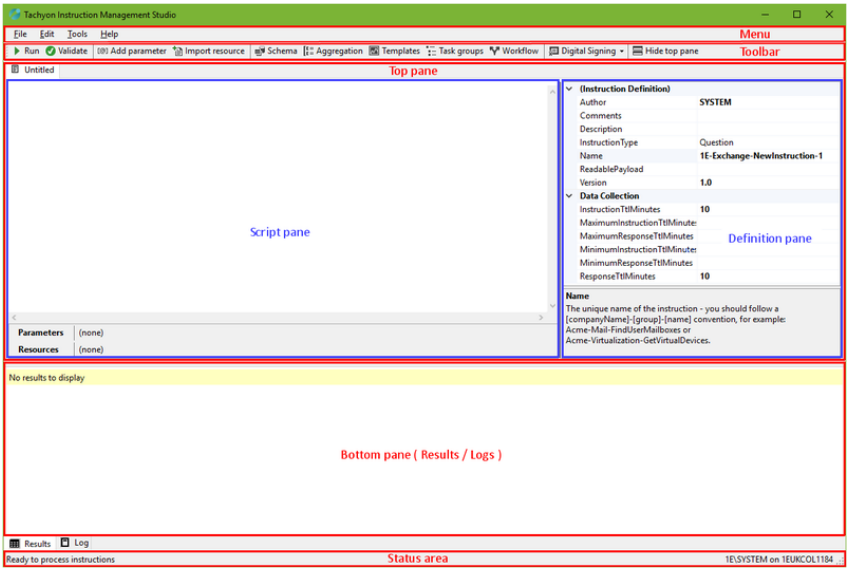
Click here to view StartTIMS.cmd ...

Download...

StartTIMS.cmd

```
:: Used to start TIMS in SYSTEM security context
:: PSEXEC.EXE should exist in local folder or a system path
:: This script should exist in the TIMS folder
:: Create shortcut and use Advanced to set as Run as administrator
@pushd %~dp0
:: create default folder for SYSTEM to avoid popups
@md "%SYSTEMROOT%\system32\config\systemprofile\Desktop" 2>nul
:: Set folder option to see file extensions
@reg add HKEY_USERS\S-1-5-18\Software\Microsoft\Windows\CurrentVersion\Explorer
\Advanced /v HideFileExt /t REG_DWORD /d 0 /f
:: Start TIMS in SYSTEM context
PsExec.exe -s -i -d "%~dp0Tachyon.InstructionManagementStudio.exe"
:: Stop and display error if Psexec not found, or not admin
@if %ERRORLEVEL% == 9009 @pause>nul
@if %ERRORLEVEL% == 5 @pause>nul
```

1.1.6 The TIMS user-interface



With the TIMS user-interface you can:

- Execute your script by clicking on Run or pressing F5
- Highlight part of your script and then execute
- Increase the size of the text in the script pane by using Zoom on your mouse (Ctrl+scroll up and down).

Results are shown in the bottom panel by clicking on the **Results** tab. You can click on any column heading to sort the results. In TIMS, columns are displayed in alphabetic A-Za-z order, not necessarily in the order defined in the instruction. The order they appear in Tachyon Explorer may be different as well and is defined by the Schema as described in Adding a schema. The Schema allows you to pick which columns are included in the response to the Server.

If you get an error, such as 'syntax error in instruction' then review the log by clicking on the **Log** tab as the bottom of the TIMS window. You can disable log scrolling by unchecking the Auto-scroll log tick box. The log will periodically scroll as the client performs background tasks, such as updating the #UUID592aa29ee0c7e3190cd79e6cd0b9cedc tables. See also Exit Codes.

You can adjust TIMS window and panel positions:

- entirely hide the top pane temporarily by clicking the 'Hide top pane' icon. Click it again to re-display the top pane
- adjust the relative size of the top and bottom panes of TIMS by dragging the horizontal splitter bar
- adjust the relative size of the script and instruction definition panes by dragging the vertical splitter bar in the top pane.

Use the F11 button to toggle visibility of the top pane (the script and instruction definition). Hiding the top pane is useful to display the bottom pane (results and log details).

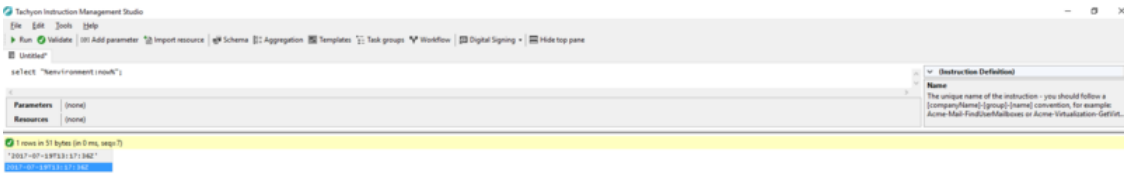
1.1.7 Quick verification that TIMS is working

1. Enter the line of text.

```
SELECT "%environment:now%";
```

2. Click the Run icon.

3. You should see the current date/time as shown below.



Note

By default, TIMS runs all the code in the script pane. However, if you select a block of code, and click Run, TIMS will run only the selected block of code, as long as it is terminated with ;

1.1.8 The TIMS menu

Above the toolbar, the TIMS menu provides additional functionality.

1.1.8.1 The File menu

The file menu allows you to:

- **Open**, **Save** and **Save As** Instruction XML file (the full definition capable of being imported into a Tachyon system)
- **Browse** folders containing Instruction XML files
- **Import** and **Export** as a *.txt file (just the script text)
- **Reset** the current instruction definition to an empty starting point
- **Exit** TIMS

1.1.8.1.1 Opening and Saving instructions

1. The **Open** option allows you to load an instruction from an existing instruction XML file. This will replace any existing script and instruction settings.
2. You will be warned if the instruction you are currently editing has any unsaved changes.
3. The **Save** and **Save As** options allow you to save an instruction XML file.
4. TIMS will validate an instruction before it is saved, as described in Validating the file.

1.1.8.1.1.1 Save warnings

When saving, you will be given various warnings asking if you wish to continue?

- You have changed the instruction in the editor without running it.

Validation warnings will be displayed before you can continue saving:

- No task groups defined.

1.1.8.1.1.2 Save errors

Validation errors will be displayed which must be corrected before you can try again.

- No Description, Name or ReadablePayload
- No schema.

Currently, TIMS does not enforce Version, but this is actually mandatory. Version should be entered as Major.Minor.

1.1.8.1.2 Importing and exporting scripts

The **Import Script** option allows you to load script text into the script pane from an existing TIMSS script TXT file. All existing schema, aggregation, workflow and parameter settings remain unchanged. You are NOT warned if the script you are currently editing has any unsaved changes.

The **Export Script** option allows you to save the script text in the script pane as a separate TIMS script TXT file.

> **Warning**
>
> In older versions of TIMS these were called TAS files, but we changed this to TXT files to avoid suggesting they were anything special.

1.1.8.2 The Edit menu

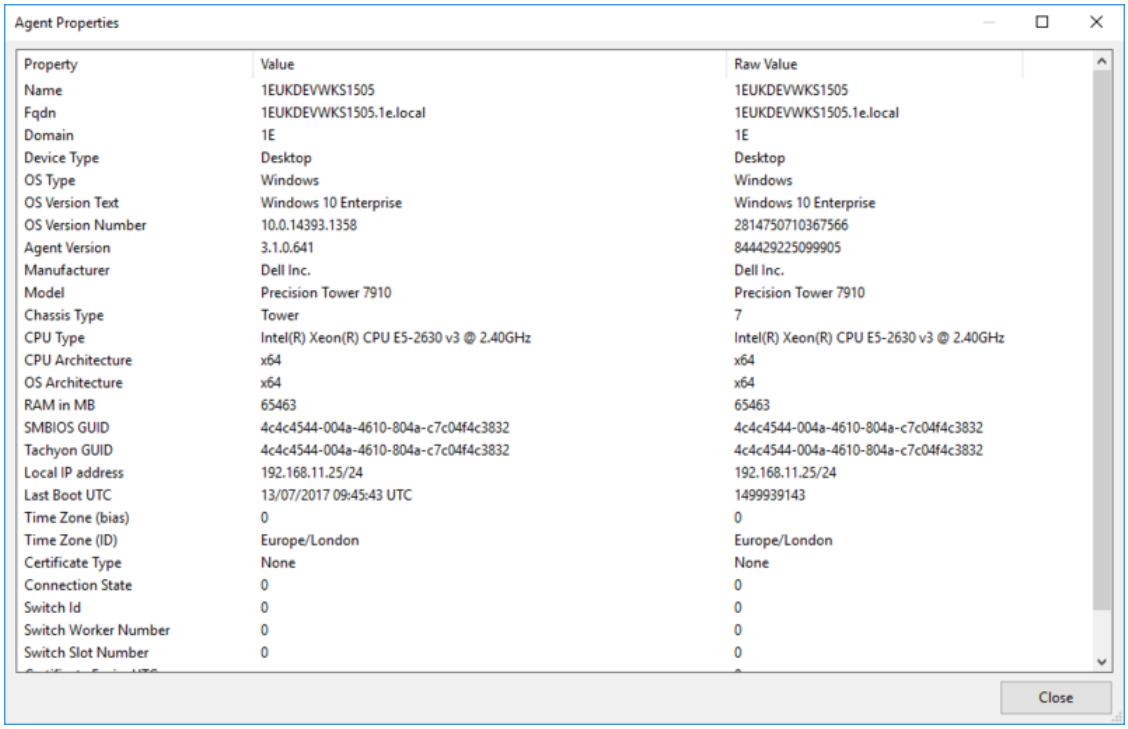
The Edit menu supports the following operations on script text in the script pane.

1.1.8.3 The Tools menu

The tools menu allows you to view the agent properties, loaded modules and change preferences.

1.1.8.3.1 Agent properties

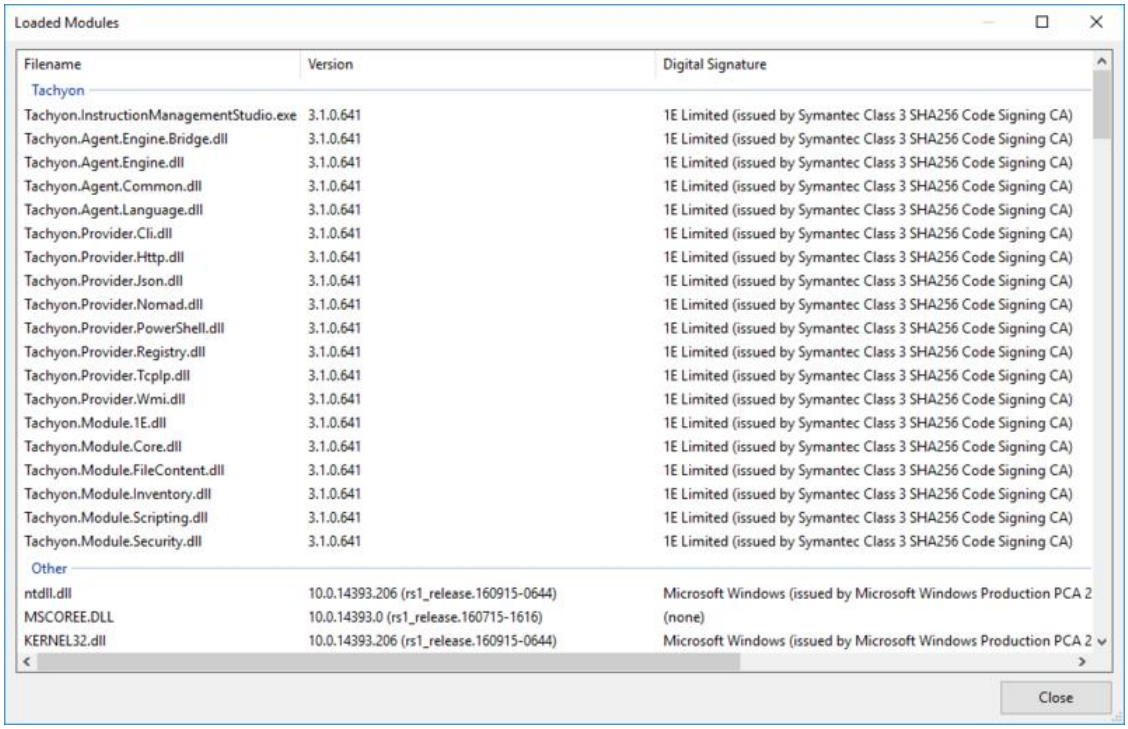
The agent properties dialog shows the current properties of the endpoint device



1.1.8.3.2 Loaded modules

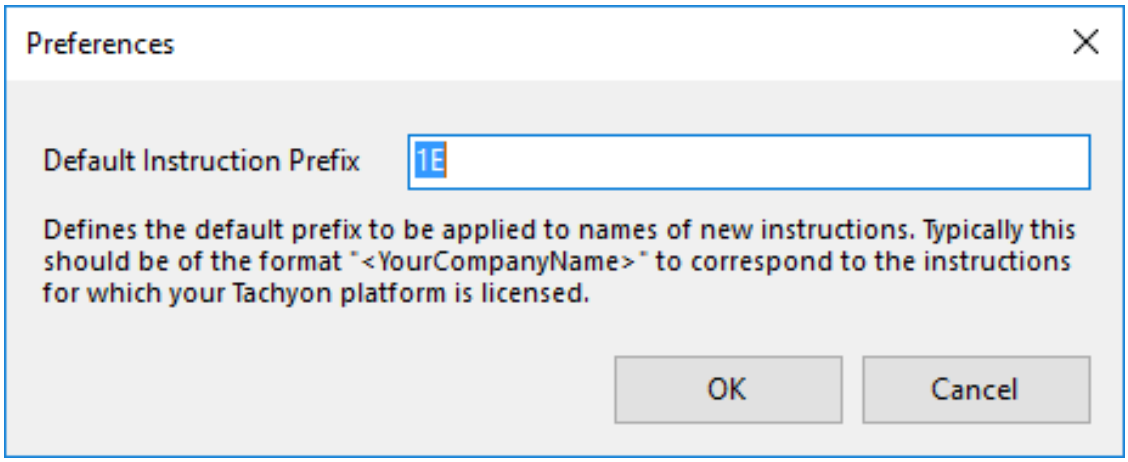
The loaded modules dialog shows all the DLLs loaded by TIMS. If you are familiar with the names of Agent language modules and methods, then you will notice the DLLs have the similar names, however some modules appear missing because they are are combined into DLLs.

As stated before, TIMS has a version of th eAgent runnig in a sanbox, so this list reflects those versions and any extensibles that may have been added.



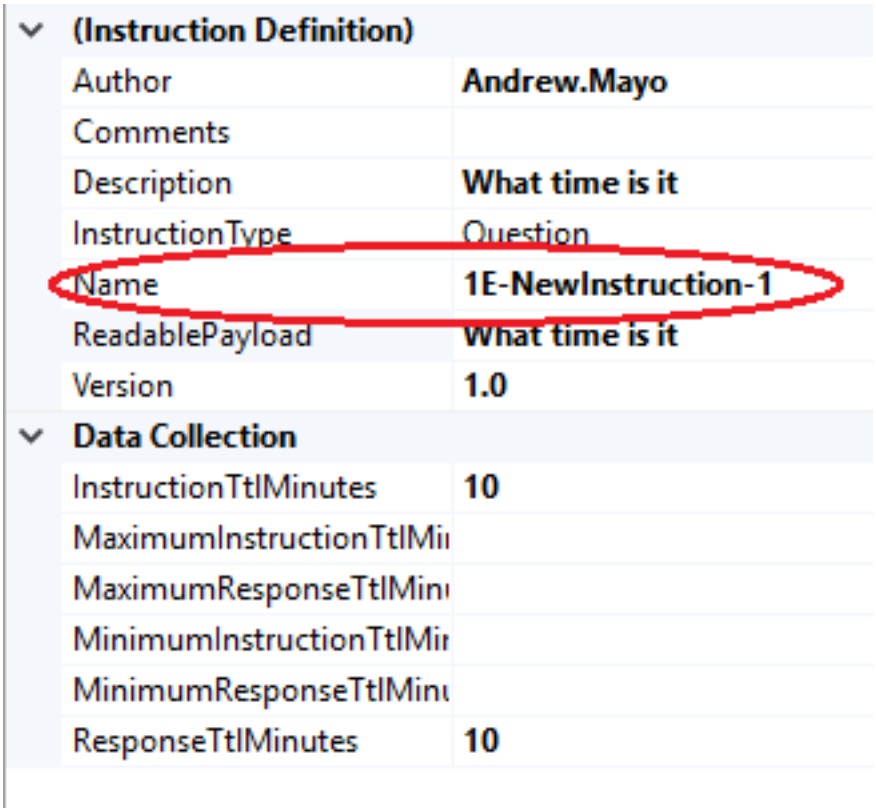
1.1.8.3.3 Preferences

The preferences dialog allows you to change the prefix associated with instruction names.



Tachyon's licensing system uses name prefixes to control which instructions can be uploaded and used. For an instruction to use a name prefix, the prefix (pattern) must be registered in your Tachyon system's license file (obtained from1E) along with details of each code signing certificate that can use that prefix. All the instructions which you sign should have the same prefix that is unique within your Tachyon system. Do not prefix your instructions with 1E.

Specifying a prefix saves time by pre-populating the Name field in the instruction definition, so that you only need to add the unique part of your instruction name.



1.1.8.4 The Help menu

The help menu displays the version of TIMS. This is usually the same as the version of the Tachyon Agent included with TIMS, but to find the actual version of the Agent go to Tools → Agent properties... → Agent version.

1.2 Using TIMS

1.2.1 Installing and starting TIMS

Please refer to Installing TIMS and Starting TIMS.

1.2.2 Creating an Instruction XML file

Instruction XML files allow code written in TIMS to be uploaded to the Tachyon server, so that the instructions can be invoked by asking a question or requesting that an action be performed.

> **Note**
>
> Instruction names must be less than 64 characters in length, otherwise 1E platform will return an error, and the instruction will not be saved.

About questions and actions

- ***Questions*** are instructions that can be searched for, and invoked, directly from the Tachyon Explorer 'I want to know' page. Questions should generally not alter the state of any endpoint on which they are run.
- ***Actions*** are instructions that can only be searched for and invoked once an initial question has been asked. You 'follow up' the initial question either by asking further questions or by invoking actions.

Note

The default instruction type for a new instruction was a question up to TIMS 5.1, but from TIMS 5.2 it is an action. This is to ensure that potentially dangerous instructions require authentications and authorisation by default. If the instruction is harmlessly "read only" then change it to a question.

Suppose that we want to create a new instruction XML file from our example above. It will perform the incredibly useful function of telling us "What time is it"

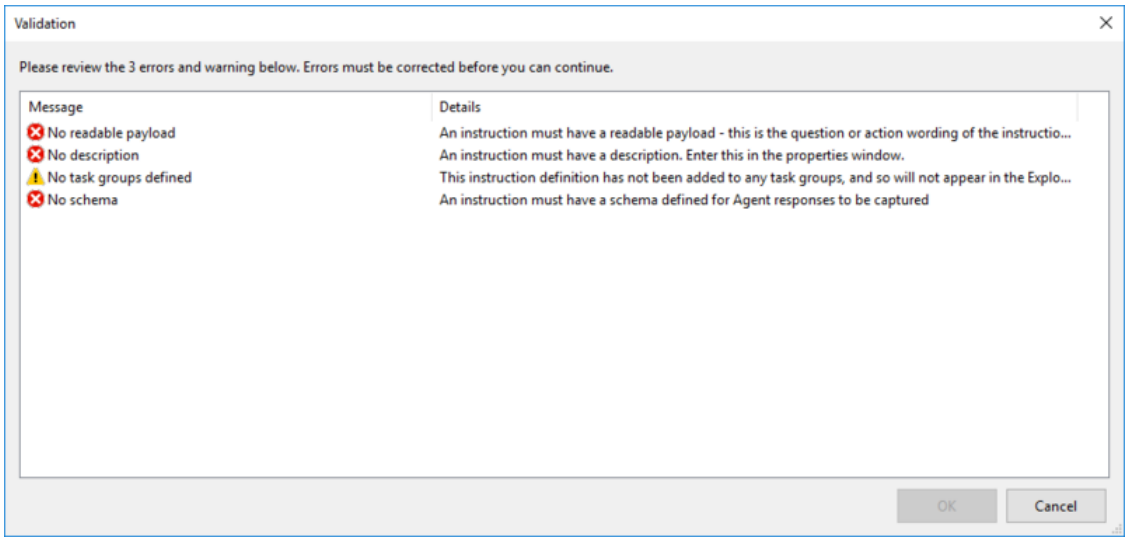
Tip

The definitions of each element of an Instruction definition are decribed in the Instruction Definition Reference. The guidance below refers to most elements but not all.

1.2.2.1 Validating the file

Select File/Save or File/Save As from the TIMS main menu.

In this case you will see a set of validation errors that need to be addressed before the file can be saved.



Note

You can also see any validation errors at any time by pressing the Validate toolbar button

1.2.2.1.1 Mandatory Settings

We will now correct the errors by adding a payload and description to the property pane.

1.2.2.1.1.1 Description

- should explain what the instruction does
- should contain useful words that will help searches
- useful to contain details such as platforms it will run on, if it downloads a script, or any restrictions

1.2.2.1.1.2 Name

- The name must be unique within your Tachyon system; you cannot load two instruction XML files into Tachyon that share a common name. If you do, the second file will overwrite the contents of the first.
- A default name has automatically been filled in
  - the first time you start TIMS the prefix is based on your Active Directory Domain Name
  - you can change the default name in Tools→Preferences

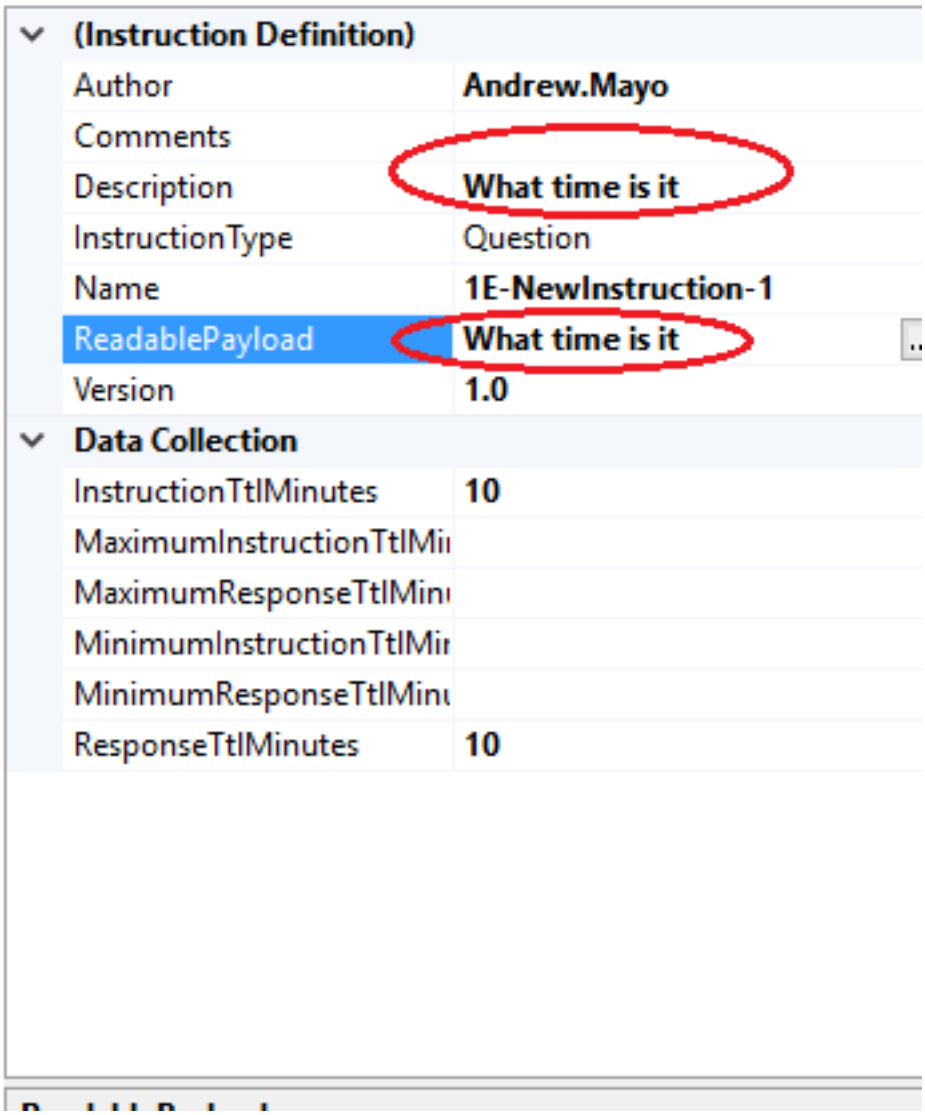
1.2.2.1.1.3 ReadablePayload

- is exactly what you see in the Tachyon Explorer when searching for questions and actions
- helps to be a grammatically correct question or action statement
- if the instruction has Instruction Parameters then these must be included so that Explorer users can see and enter the parameters.

1.2.2.1.1.4 Version

- can be a single digit or a dotted notation with 2, 3 or 4 parts.

When using Tachyon Explorer to search for instructions, the search will parse the ReadablePayload and Description fields.



Tip

See Instruction Definition Reference (InstructionDefinition) for descriptions of the other settings (elements).

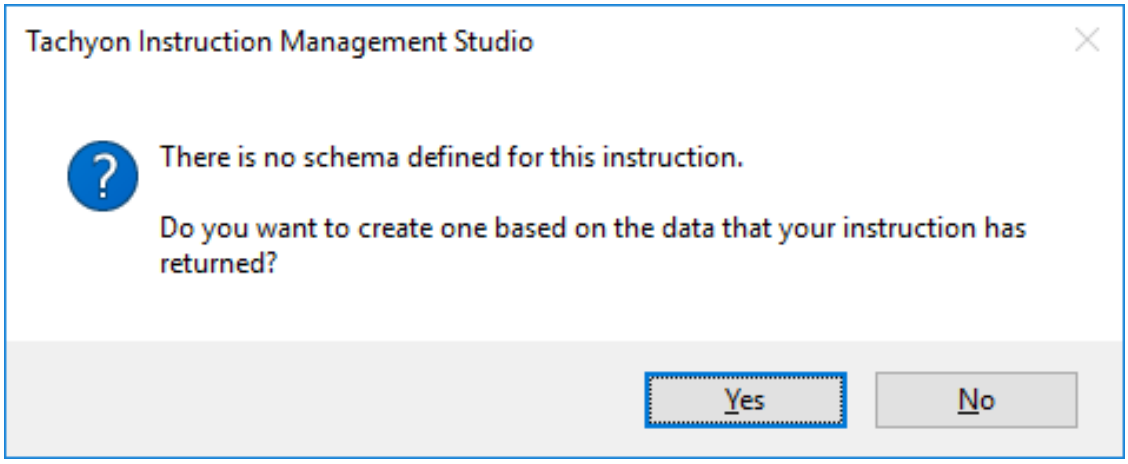
1.2.2.2 Adding a schema

The schema defines how the instruction output is actually displayed in Tachyon Explorer.

You must define the schema, or format, of the information that will be displayed in Tachyon Explorer, which may be different to what is actually output by the instruction and returned by the Tachyon client to the Server. You would not normally want to send back more data than you want to display.

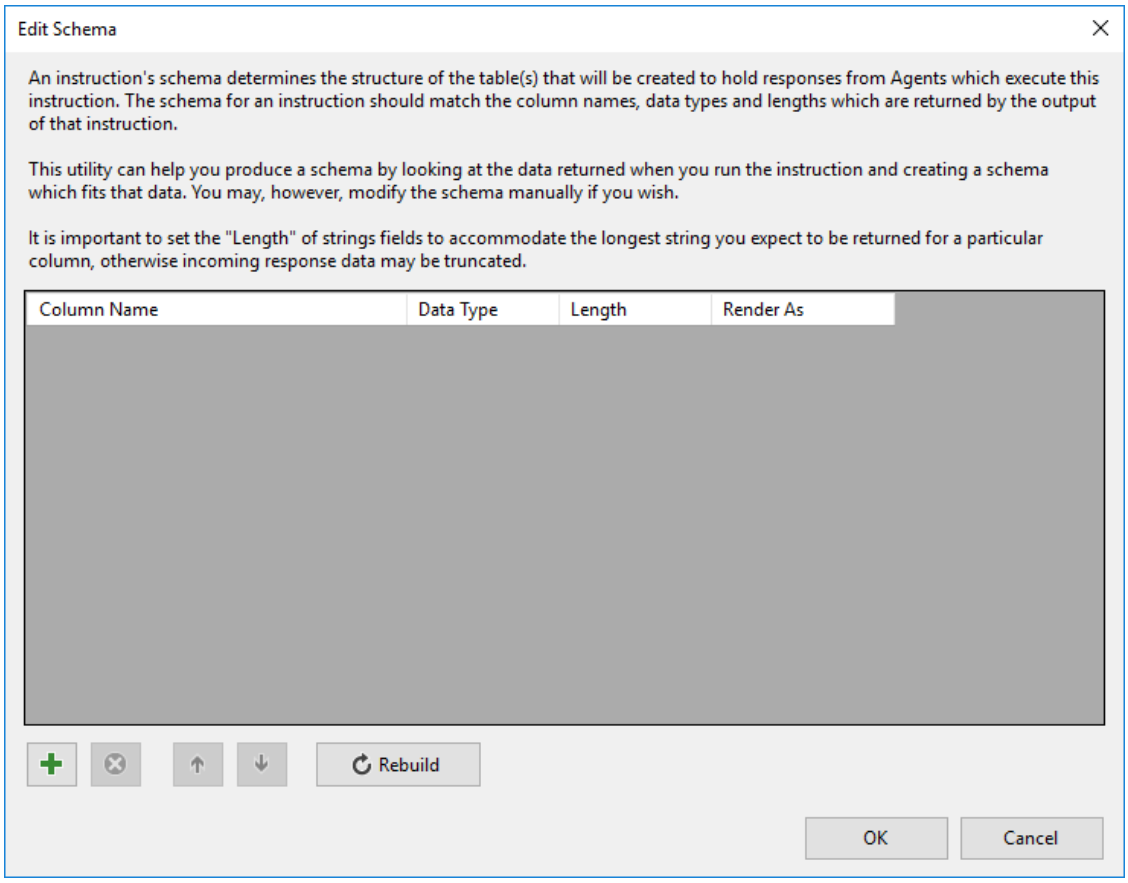
The schema also defines how the data will be rendered. Normally, you would code your instruction so that its output column names are the same as you want to display. However they may be circumstances when you prefer to use the schema definition to change column names, or reformat or render in a different way.

Press the schema button to define or modify the schema.



TIMS can infer a schema based on the data returned by your instruction, or you can manually define one.

If you select No in the above dialog you will be presented with an empty schema dialog like this.

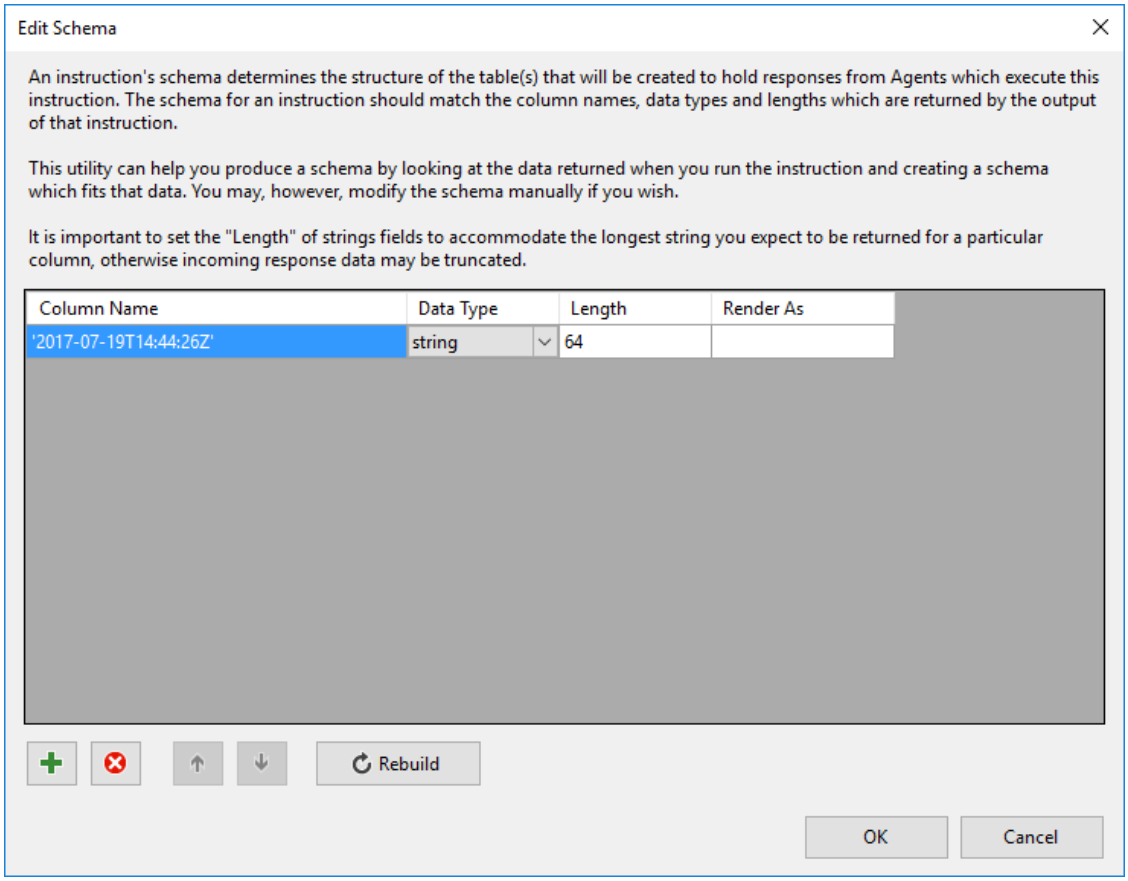


Use the + button to add a new schema entry, specifying the column name, data type and (optionally) the length

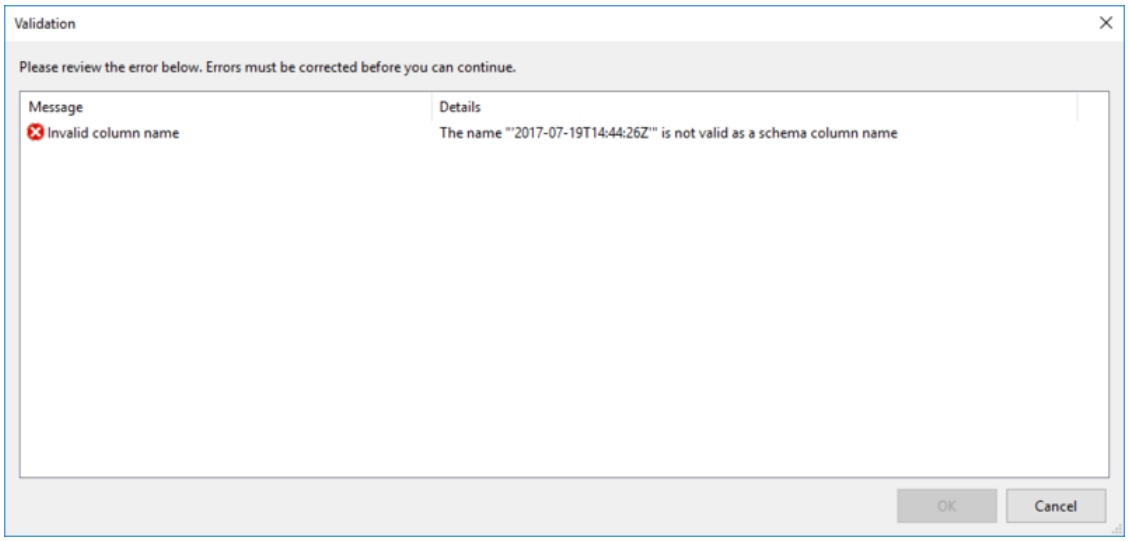
Note

The 'render as' field is an optional field and will not be used in this example.

If you select Yes in the above dialog, TIMS will attempt to infer a schema by examining the results that running the instruction produced



However in the latter case, note that the column name is not valid. You will be notified of this when you try and confirm the schema by pressing OK



1.2.2.2.1 Alter the column name to a valid name

Alter the column name to 'currenttime'. You will now be able to save the schema

1.2.2.2.2 The 'render as' schema item property

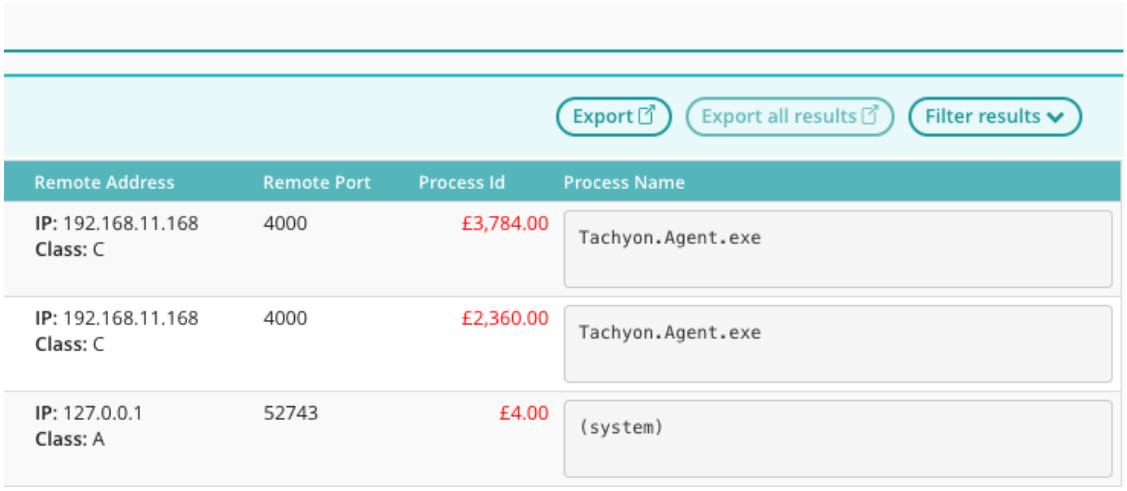
This field allows you to select a rendering scheme which will be applied to the column when it is displayed in the Tachyon Explorer.

There are three rendering schemes currently available

- preformatted
- ipAddress
- currency

The screenshot below shows the effect of each of these.

- The Remote Address column has been associated with the **ipAddress** rendering scheme.
- The process ID column has been associated with the **currency** rendering scheme.
- The Process Name column has been associated with the **preformatted** rendering scheme



Tip

See Instruction Definition Reference (SchemaJson) for more detail about Schema.

1.2.2.3 Adding a task group

You will receive a warning if you try and re-save the file, that you have not associated this instruction XML file with any task group in Tachyon. Task groups allow you to associate instructions with specific tasks in Tachyon, which are visible from the Tachyon Explorer task pane.

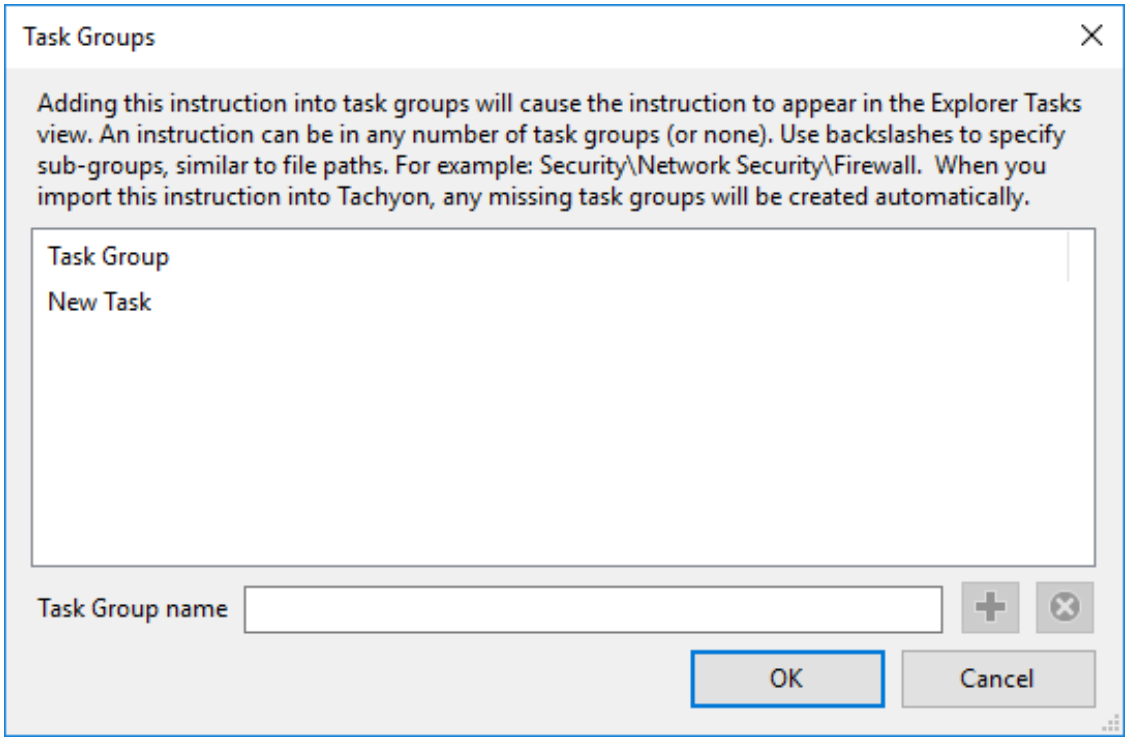
It is not mandatory to associate an instruction XML file with a task group.

If you wish to do so, click the Task groups button on the TIMS toolbar

Enter a task group name and press the + button to add the task group name.

Note

You can enter multiple task groups, and you can also create task group hierarchies, by separating each part of the task group name with a backslash (\).



Press OK to save changes

Tip

See Instruction Definition Reference (TaskGroups) for more detail about Task Groups.

1.2.2.4 Saving the instruction XML file

You have now addressed all the validation errors and warnings and have a valid instruction XML file.

Select Save or Save As and specify the file name if required.

1.2.2.5 Examining the XML file

This is what the generated XML file looks like, at this point

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<InstructionDefinition Author="Andrew.Mayo" Name="1E-NewInstruction-1" Readable
Payload="What time is it" Description="What time is it" InstructionType="Questi
on" InstructionTtlMinutes="10" ResponseTtlMinutes="10" Version="1.0" xmlns="htt
p://schemas.1e.com/Tachyon/InstructionDefinition/1.0">
  <Payload><![CDATA[SELECT "%environment:now%";

]]></Payload>
  <SchemaJson><![CDATA[[
  {
```

```
    "Name": "currenttime",
    "Type": "string",
    "Length": 64
  }
]]]></SchemaJson>
  <TaskGroup>
    <TaskGroup Name="New Task" />
  </TaskGroup>
</InstructionDefinition>
```

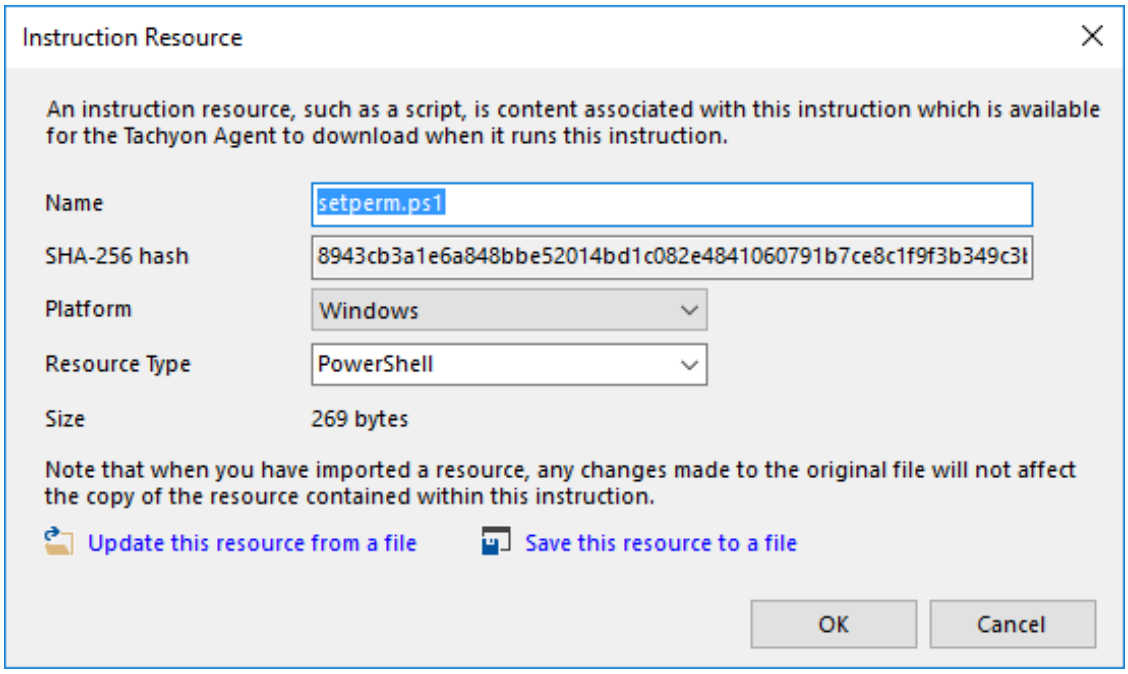
1.2.3 Adding resources

Resources are additional files that are required when executing instructions in an instruction XML file. Resources are downloaded from the Tachyon content URL by the Tachyon client, when they are accessed.

TIMS embeds resources directly into the instruction XML file, so that when this is loaded into Tachyon, the embedded resources are automatically extracted and copied to the Tachyon content URL.

To add a resource, click the **Import Resource** toolbar in the TIMS menu. Browse to the file and select it.

TIMS will automatically determine the resource type and compute its SHA-256 hash and size as shown in the dialog below



The options 'update this resource from a file' and 'save this resource to a file' will be discussed shortly. They are mainly useful when managing already-embedded resources.

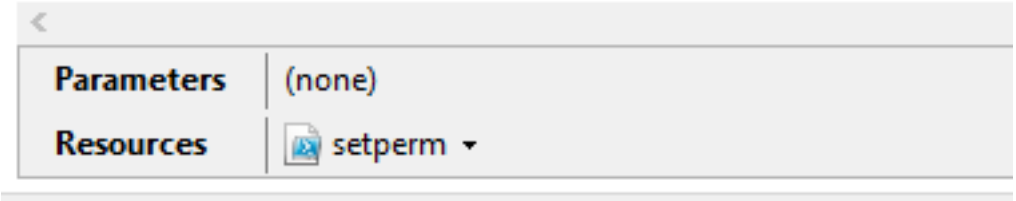
Press OK to insert the resource. The code block below shows how the instruction XML file looks after the resource has been embedded. Note that the base-64 encoding of the resource has been truncated for clarity.

```
<Resources>
    <Resource Name="setperm">
      <Content Type="PowerShell" FileName="setperm.ps1" Size="269" Platform="Wi
```

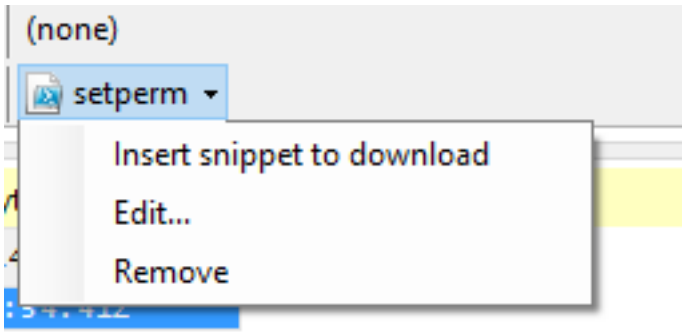
```
ndows" Hash="8943cb3a1e6a848bbe52014bd1c082e4841060791b7ce8c1f9f3b349c3bd8b64">
                <![CDATA[JGFjbD1nZXQ.................]]>
          </Content>
    </Resource>
  </Resources>
```

1.2.3.1 Managing embedded resources

Once a resource has been embedded, TIMS shows it in the script pane footer, like this



Click on the resource dropdown to see the available resource management options



1.2.3.2 Insert Snippet to Download

The Insert Snippet to Download option inserts code to download the resource programmatically. For example, for the PowerShell resource we embedded, the snippet will look like this

```
@resource1 = HttpGetFile(URL:"setperm.ps1", Size:269, Hash:"8943cb3a1e6a848bbe5
2014bd1c082e4841060791b7ce8c1f9f3b349c3bd8b64");
```

1.2.3.3 Resource downloading in the TIMS environment

When your instruction is deployed in the Tachyon environment, the above HttpGetFile request will download the resource file from the Tachyon server content endpoint URL.

However while you are developing your instruction in TIMS, when the HttpGetFile is executed, TIMS will instead retrieve the embedded resource from its internal memory and populate the result variable, just as if the file had actually been downloaded.

1.2.3.4 Edit

The Edit option will display the resource dialog we saw previously. You can amend any resource property from this dialog.

You can also select the two options which were mentioned previously.

1.2.3.4.1 Update this resource from a file

When this option is selected, you will be prompted to select a file. The file contents completely replace the current resource.

1.2.3.4.2 Save this resource to a file

When this option is selected, you will be prompted to enter a file name. A copy of the resource is saved to the file you specify

1.2.3.5 Remove

The resource is removed from the instruction XML file.

> **Tip**
>
> See Instruction Definition Reference (Resources) for more detail about resources.

1.2.4 Instruction Parameters

Our example instruction did not require any input parameters. For instructions that require parameters, TIMS allows you to define each parameter, along with its type, and various other attributes.

If you specify a parameter in TIMS, you will be prompted to enter a parameter value each time you run the instruction. The previous entry entered during that editing session is remembered.

Let's change our example to accept a single parameter.

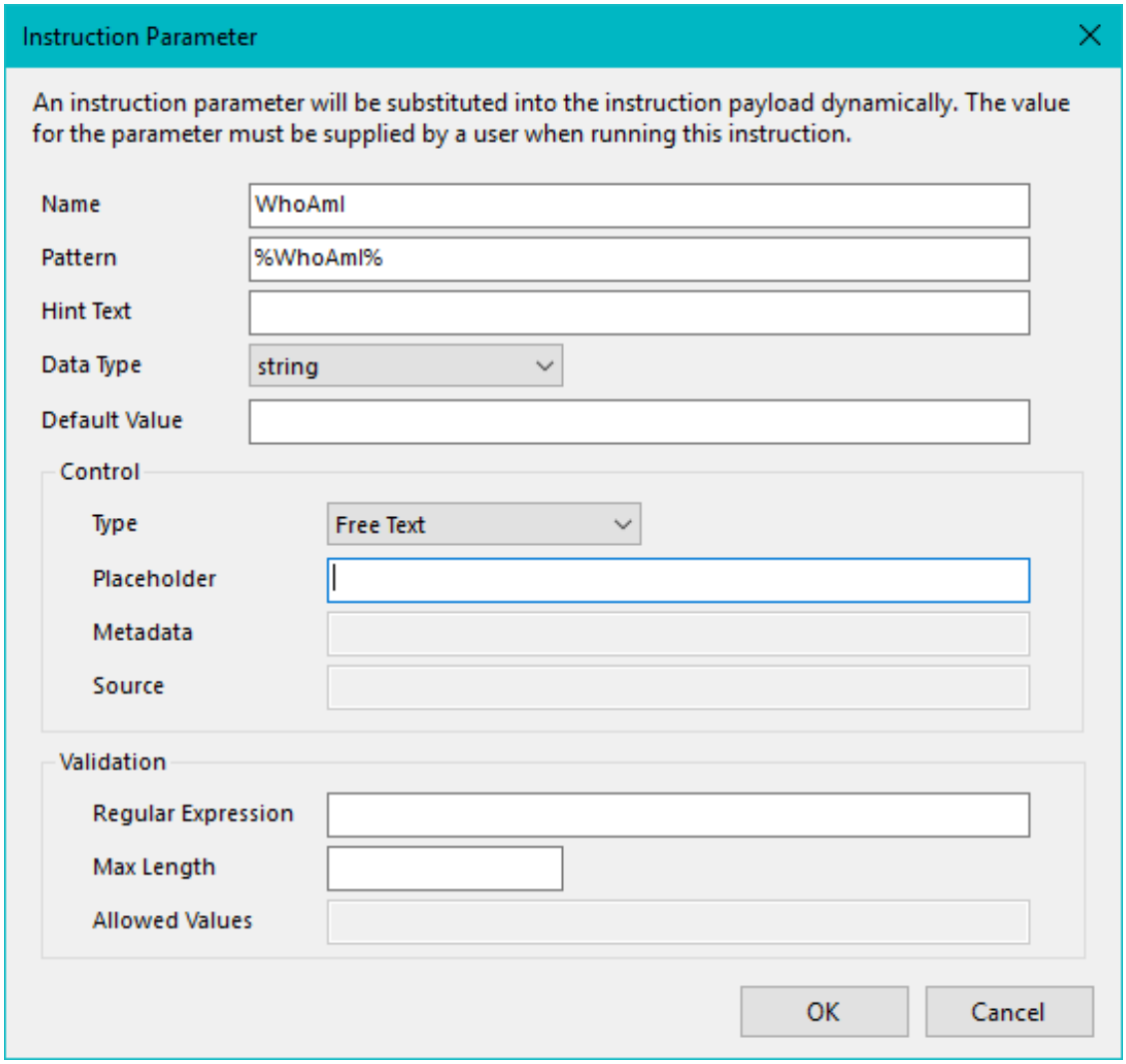
Remove the existing instruction and then click the **Add Parameter** toolbar on the TIMS menu

Enter a parameter name **WhoAmI**

The parameter pattern - by which it will be referred to in code - is automatically inserted for you, wrapped by the required % symbols.

Field	Description
Name	Parameter name - must be unique in the instruction
Pattern	When the instruction is run, this text will be substituted in the instruction payload and readable payload with the parameter's value. The text must be enclosed by % symbols and should be the same as the unique paramter name.

Field	Description
Hint Text	The alternative text to show (instead of the name) in the readable payload.
Data Type	• string • int • float • date • time
Default Value	default value for the parameter.
Control Type	• Free Text • Value Picker • Cutsom Property Key • Custom Property Value
Control Placeholder	The text to show as a prompt to the user if th econtrol s empty. Valid for Free Text only.
Control Metadata	Ths must match an existing custom property type (e.g. "Coverage Tag"). Valid only for Custom Property Key and Value.
Control Source	The parameter name of the corresponding custom property key. Valid only for Custom Property Value.
Validation Regular Expression	Regular expression, in .NET Framework regular expression syntax. Valid only for string Free Text fields.
Validation Max Length	Maximum number of characters. Valid only for string Free Text fields.
Validation Allowed Values	List of allowed values, separated by commas. Valid only for Value PIcker entry type.



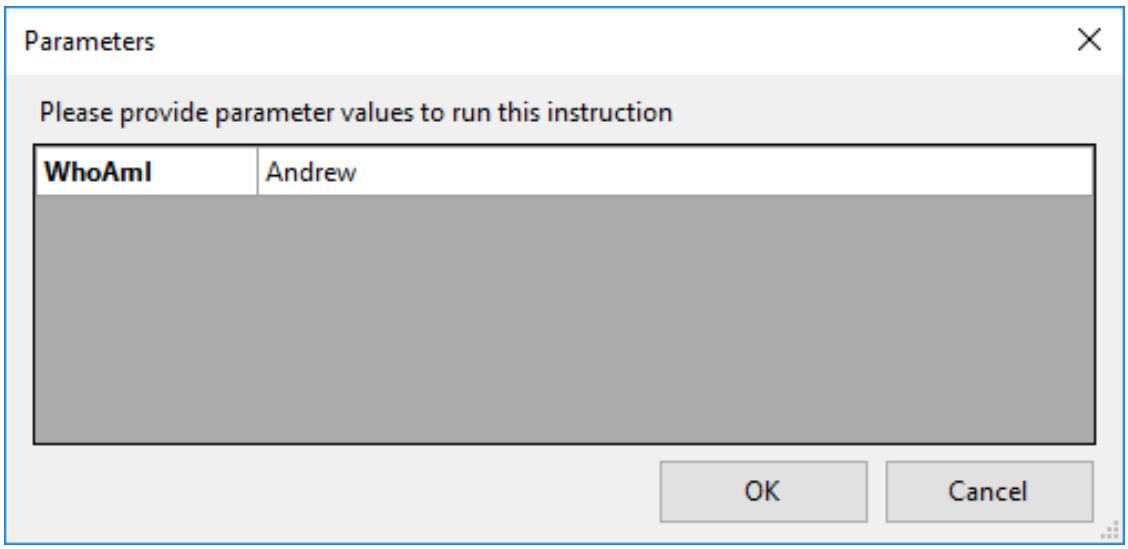
Note

In order for a user to enter parameter values when running an instruction in Explorer, the Readable Payload must include the same parameter names. For example:
Say hello to %WhoAmI%

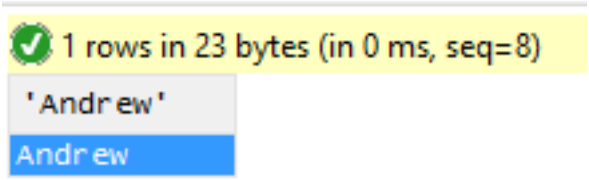
Add an instruction in the top pane of TIMS

```
select "%WhoAmI%";
```

Run the instruction. You are prompted for a parameter value.



Enter a parameter value and press OK. The instruction runs and the result is displayed.



Note

Parameter patterns are case-sensitive. %whoami% would not be the same parameter as %WhoAmI%

Instruction parameter considerations

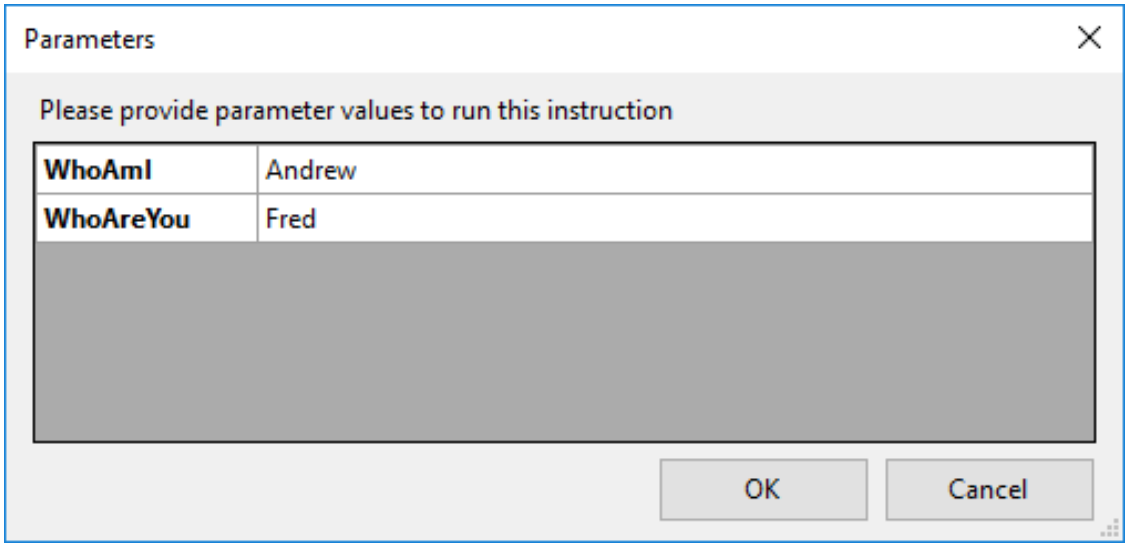
Since parameter values are substituted directly into the Agent's payload, you must take care to avoid the possibility of SQL or statement injection. Tachyon takes precautions against this (e.g. escaping double-quotes and backslashes within string parameter values), but you should consider the following when parameterizing instructions:

- Although the format %ParameterName% is the preferred convention for a parameter placeholder, you are free to use any substitution placeholder you wish
- Parameters should be defined as the most appropriate data type (string, integer, float, Boolean)
- String parameters should, where possible, be constrained using a regular expression to prevent invalid input
- Placeholders for string parameters should be placed within double-quotes within the Agent Language – the Tachyon Consumer API will automatically perform escaping of the parameter's value
- Placeholders for non-string parameters should not be placed in double-quotes
- Within SQL statements, you may wish to consider using a CAST("%parameter%" AS NUMBER) expression for type-safety

- Although it is possible to use parameters to build "dynamic statements" within the Agent Language, it is NOT recommended unless you take extreme care (and limit the possible values which can be supplied for the parameter)

1.2.4.1 Adding multiple parameters

If you add more parameters, then when you run an instruction you see all parameters and their current values in the parameters dialog, like this



1.2.4.2 Managing parameters

As with resources, parameters that have been added are shown in the footer pane of the script pane, like this

Click on a parameter dropdown to see the parameter menu.

1.2.4.2.1 Insert substitution pattern

This option just inserts the parameter pattern (such as %WhoAmI%) in the script pane.

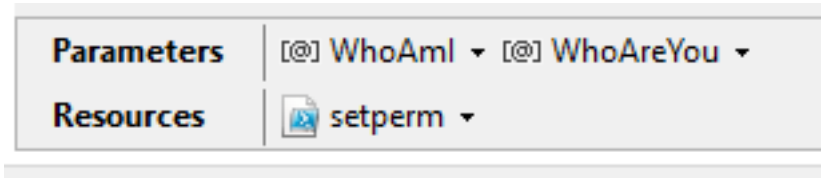
1.2.4.2.2 Edit...

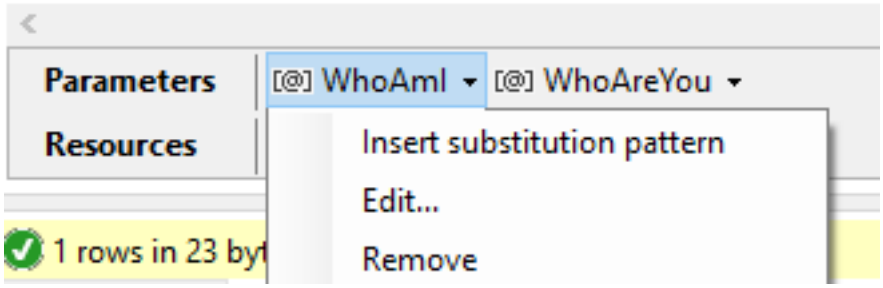
This option brings up the parameter dialog which we saw earlier, and allows you to change any property of the parameter

1.2.4.2.3 Remove

This option removes (deletes) the parameter. Any references to the parameter in instruction code will also be removed, and replaced with the current value of the parameter. For example, if I remove the %WhoAmI% parameter my code changes to

```
select "Andrew";
```





1.2.4.3 Parameter properties

The parameter dialog shows a number of parameter properties which control how it appears in the Tachyon Explorer along with default values, validation patterns and so forth. Because each field has a descriptive tooltip that appears when you hover over it, I will not repeat this information in the documentation here.

> **Note**
>
> In earlier versions of TIMS the parameter properties dialog includes a field for Numeric Restrictions. It is ignored. This feature does not exist yet. This field is removed from later versions of TIMS, pending support for this feature in a future version of Tachyon.

> **Tip**
>
> See Instruction Definition Reference (ParameterJson) for more detail about parameters.

1.2.5 Aggregation

Instruction results can be automatically aggregated by Tachyon, so that when a user clicks on an aggregated result, Tachyon then 'drills down' and displays the data that was aggregated.

To aggregate results you need to return at least two columns from the instruction.

One column will be used to group the results. Results in each group are then aggregated by one or more of the remaining columns in the result set. Aggregation means computing some value across all the results, such as their sum, min, max etc.

> **Note**
>
> You can group results by multiple columns as well. However in our example we will group by only one column to keep things simple.

We will modify the instruction schema and instruction that we have been working with in order to set up aggregation.

> **Tip**

See Instruction Definition Reference (AggregationJson) for more detail about Aggregation settings.

Note

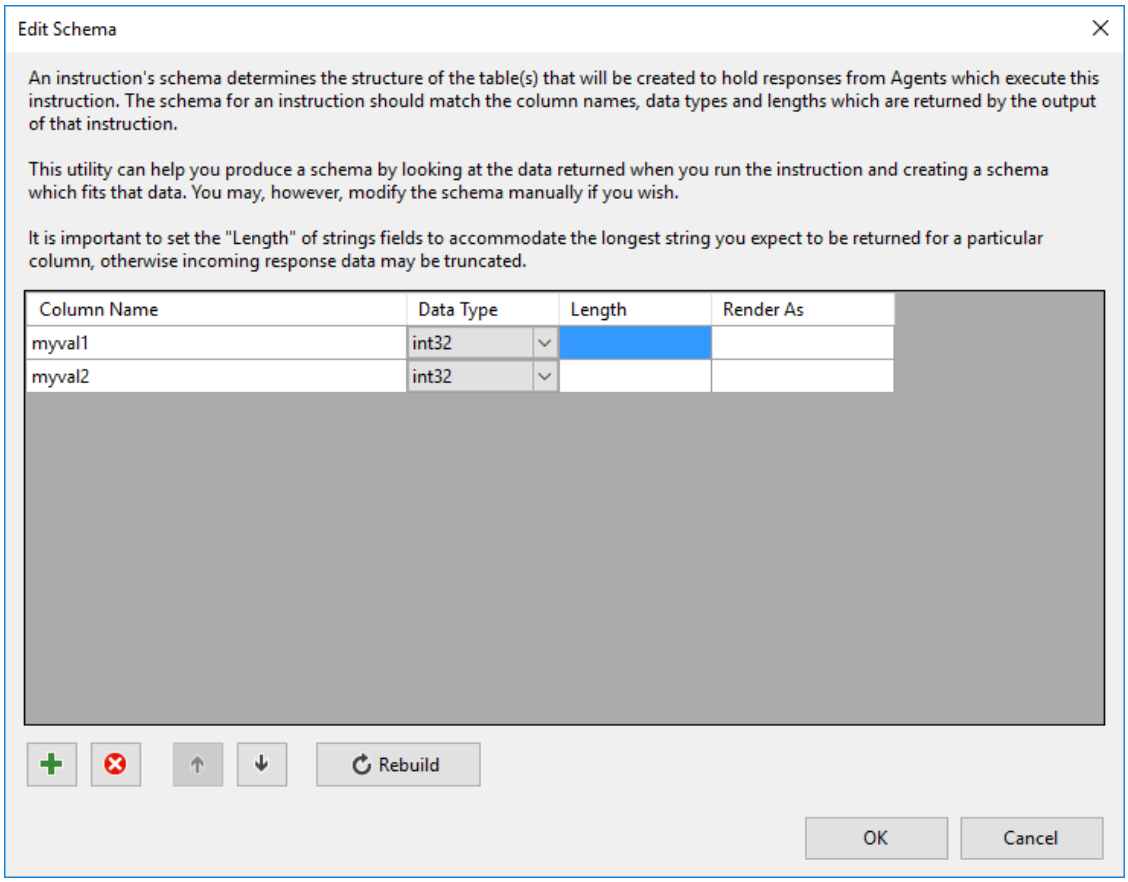
TIMS allows you to author instructions which use aggregation but it cannot display the aggregated result nor does it support the ability to drill down. You will have to upload the instruction to Tachyon in order to utilize aggregation. When run under TIMS, only the raw (non-aggregated) results are displayed.

1.2.5.1 Configure Aggregation

Press the schema button and modify the schema to return two columns, myval1 and myval2 of type int32

Note

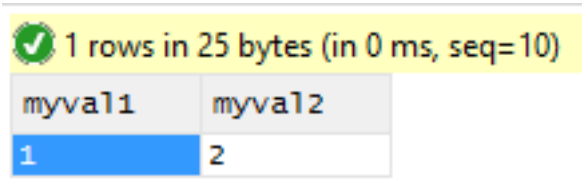
Use the + button to add additional columns to the schema.



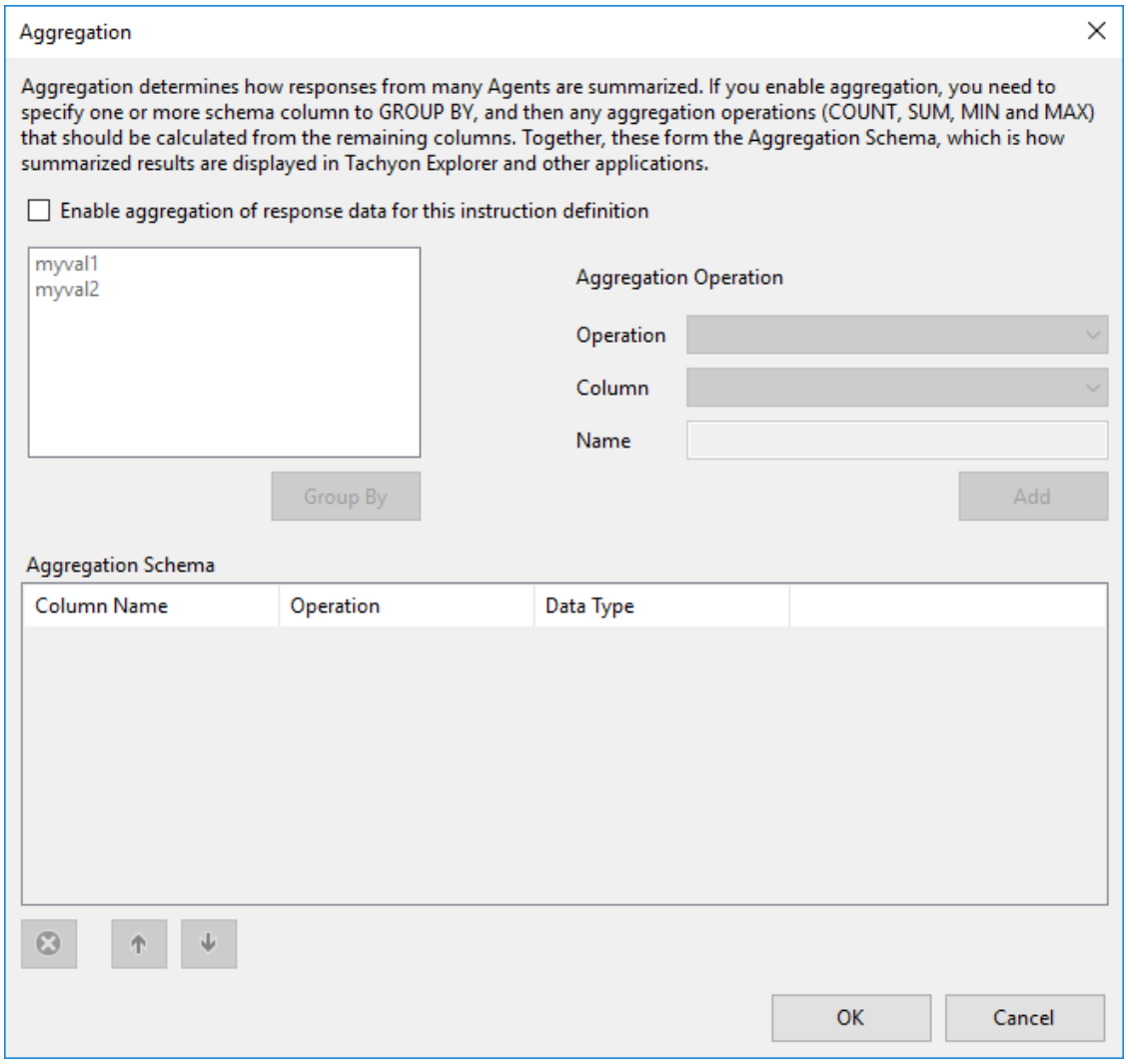
Press OK and then modify the instruction to read

```
select 1 myval1, 2 myval2;
```

Run the instruction and observe that values are returned for the two columns



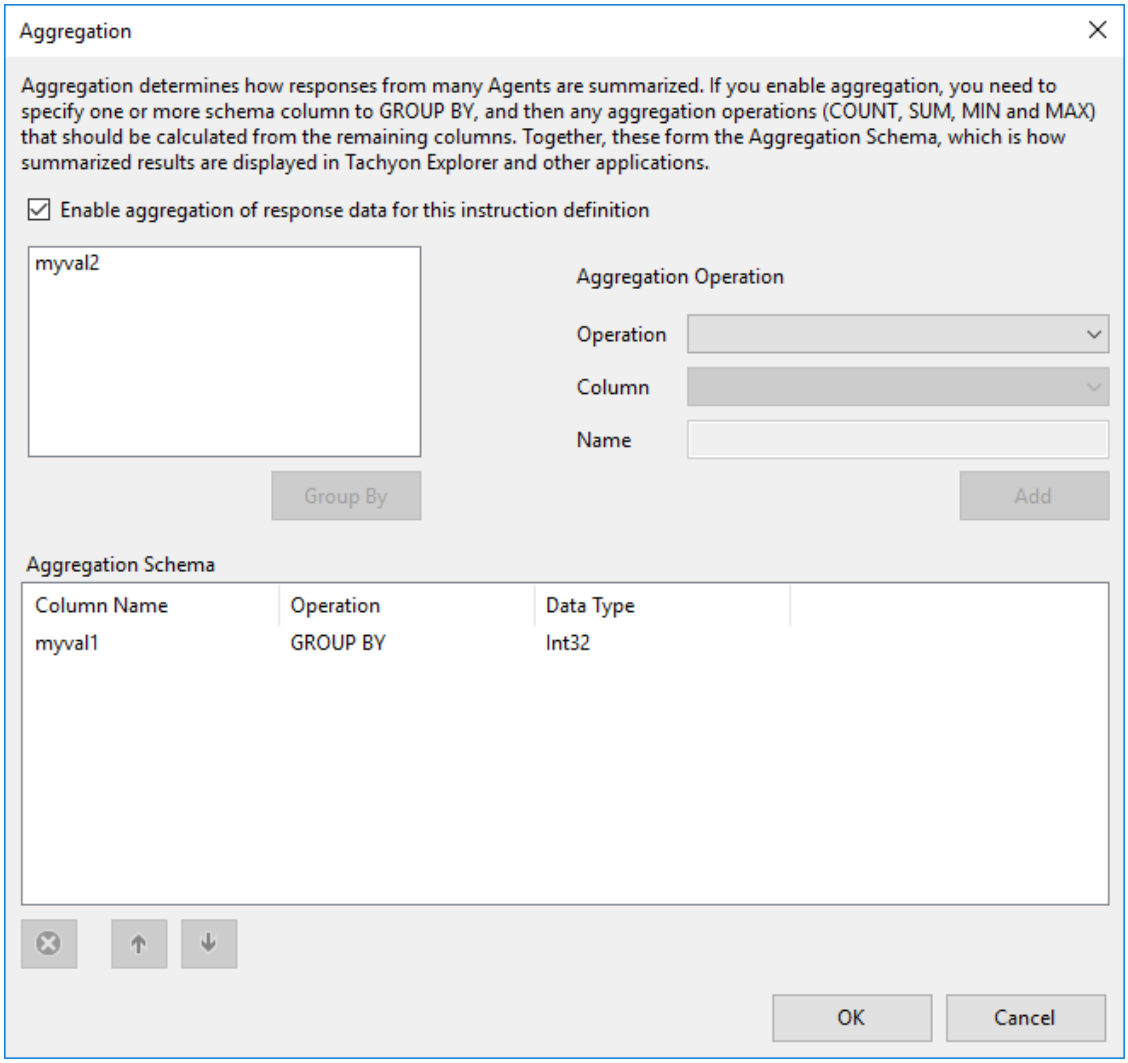
We will now define an aggregation for the results. This will control how the Tachyon server processes the raw results from each endpoint when this instruction is executed



1.2.5.1.1 Enable aggregation and select a 'group by' column

First you must tick the 'Enable aggregation checkbox

Then select the column you wish the results to be grouped by on the server side. In this case we have selected that the results should be grouped by the 'myval1' column, by clicking the column name and then clicking the Group By button

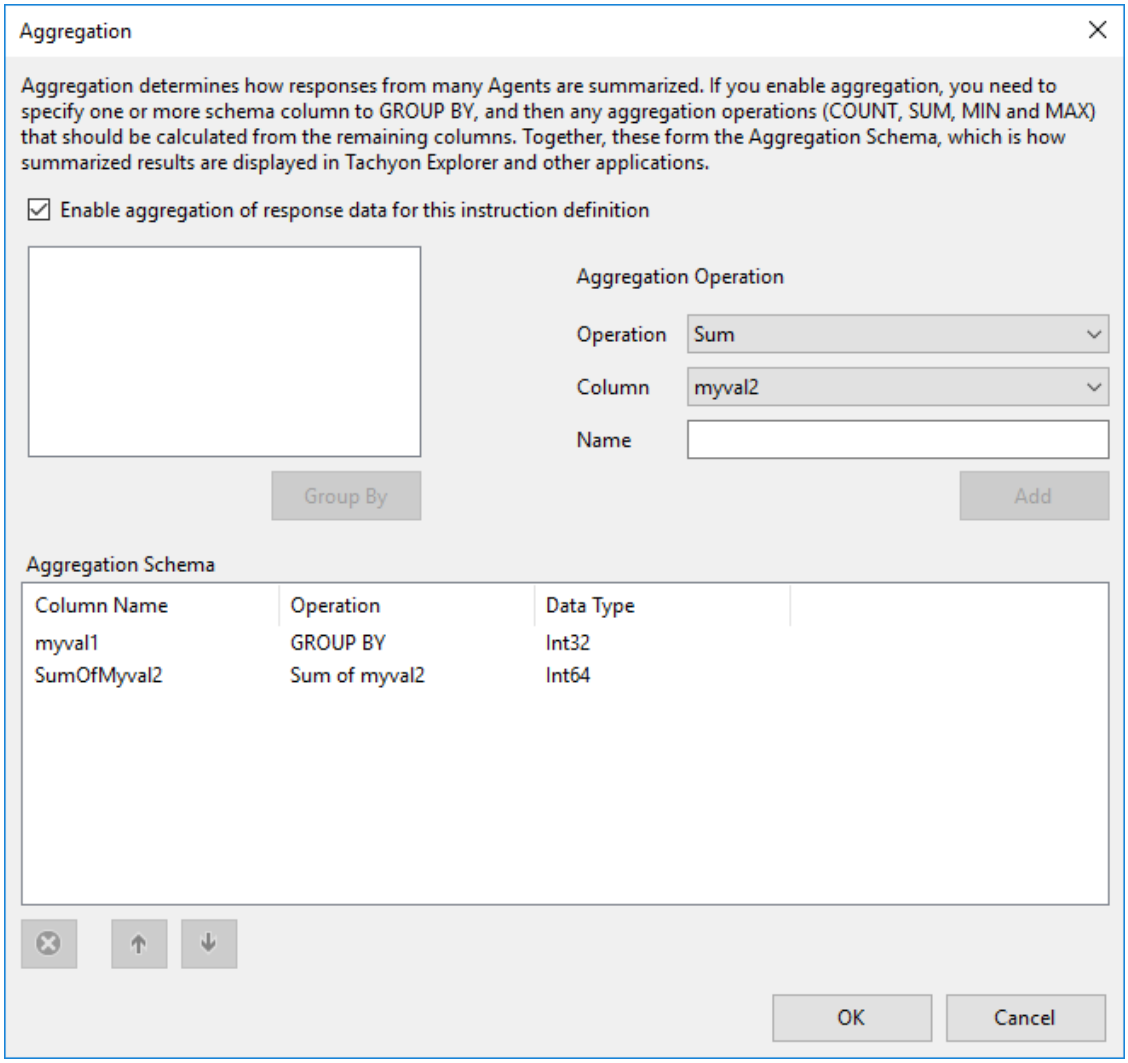


1.2.5.1.2 Choose an aggregation operation and a target column

Choose 'sum' from the operation dropdown list and choose the myval2 column from the column dropdown list

We will name the result 'SumofMyval2'

Then click Add to add the aggregation operation



Press Ok to save changes

The aggregation data is added to the instruction XML file. It will look like this (note: the XML has been reformatted for clarity)

```
<AggregationJson>
        <![CDATA[
                        {"Schema":[
                                        {"Name":"myval1","Type":"int32"
,"Length":"0"},
                                        {"Name":"SumOfMyval2","Type":"i
nt64","Length":"0"}
                                ],
                                        "GroupBy":"myval1","Operations"
:[{"Name":"SumOfMyval2","Type":"sum(myval2)"}]}
        ]]>
</AggregationJson>
```

When data is collected by the Tachyon client, the results will be grouped by the value of the myval1 column for all devices that respond. Then, the myval2 column will show the sum of the

values of each device's myval2 column results, for that particular grouping value returned in myval1. For example:

Raw data collected from all endpoints

Device	myval1	myval2
D1	1	2
D1	1	3
D2	1	4
D2	2	1
D3	2	5

Aggregated data

myval1	SumofMyval1
1	9
2	6

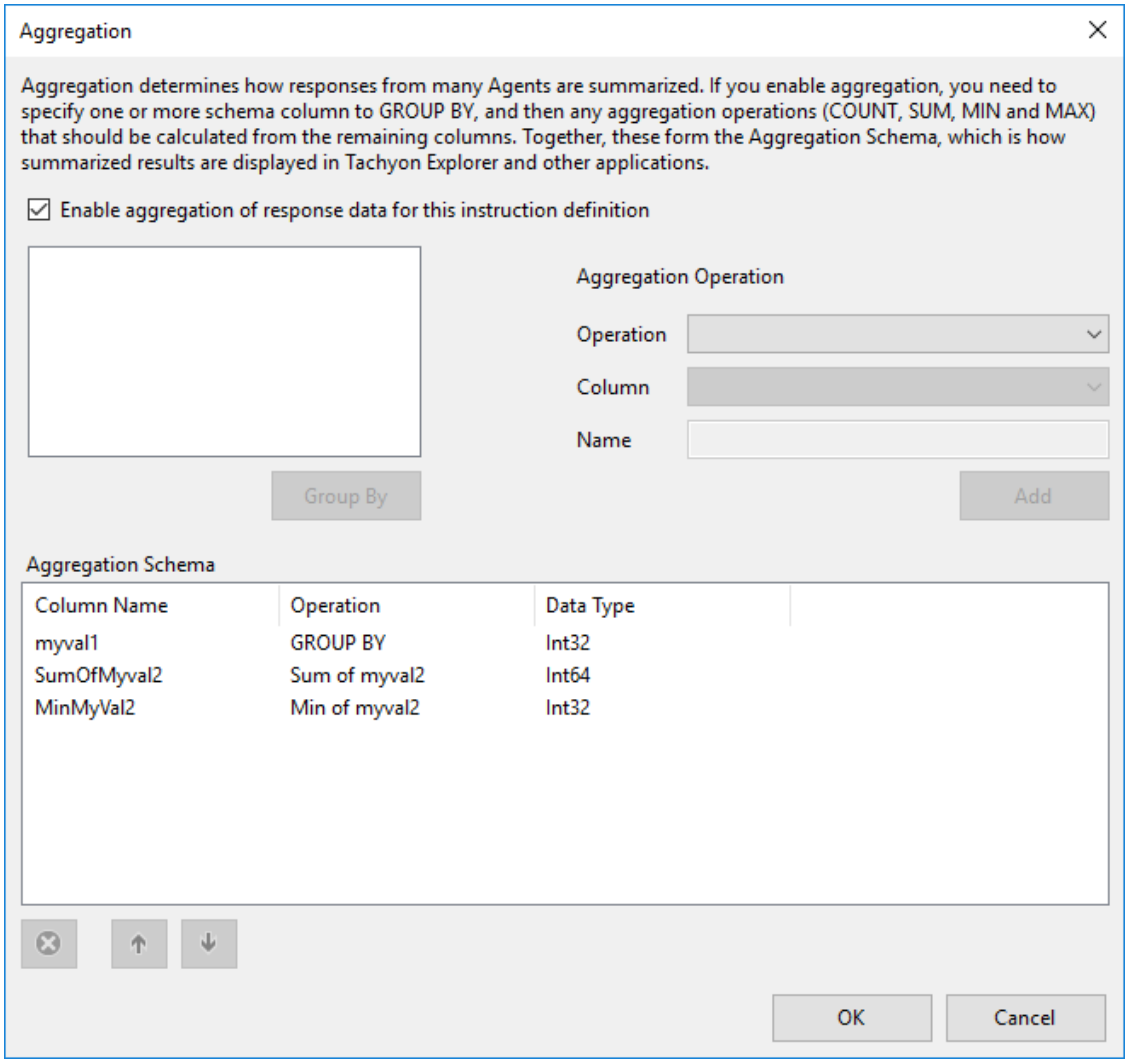
You can then 'drill down' by clicking on any aggregated row to view the raw data that was aggregated to produce the results.

1.2.5.2 Computing multiple aggregated results

Just as you can group by multiple columns, you can compute aggregated results multiple times, either for the same column or for different columns.

These columns cannot be part of the aggregate grouping columns, however.

In the example below we compute both the sum of the myval2 columns returned and the minimum value of the myval2 columns.



1.2.5.3 Enabling and disabling aggregation

The 'enable aggregation' checkbox in the aggregation dialog allows you to disable aggregation at any time. However, note that if you do this, your current aggregation settings are immediately lost when you close the dialog.

1.2.6 Workflow

Tachyon questions and actions are associated with workflow definitions which define how the questions and actions are processed when they are submitted.

There are two global workflow settings which are used by default.

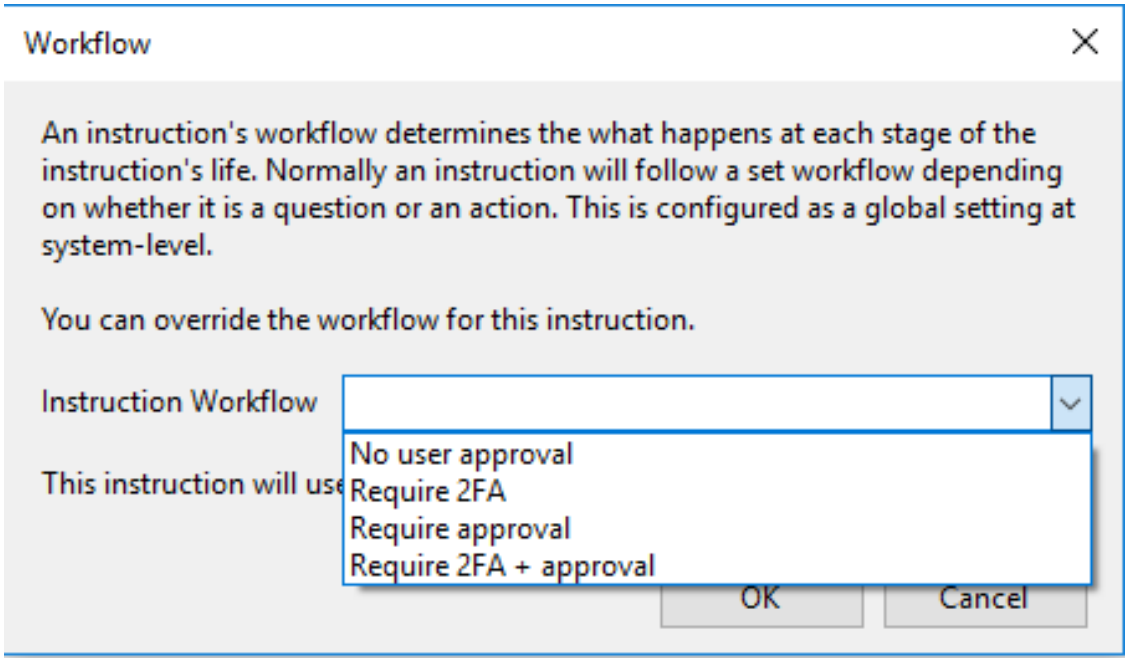
One global workflow setting is associated with questions, and one with actions.

You can override the global defaults for an instruction by clicking the workflow button in the toolbar

Choosing the empty drop-down option means that the default globally configured workflow option will be used when the instruction is run. The global default will depend on whether the instruction is a question or action

Otherwise, choosing an alternative workflow option will override the global default for the instruction.

Workflow options are discussed in more detail in the documentation on Tachyon Workflow



Note

For potentially state-altering actions, use caution in overriding the default workflow.

If workflow is overridden, the instruction XML file will contain a workflow section, such as this

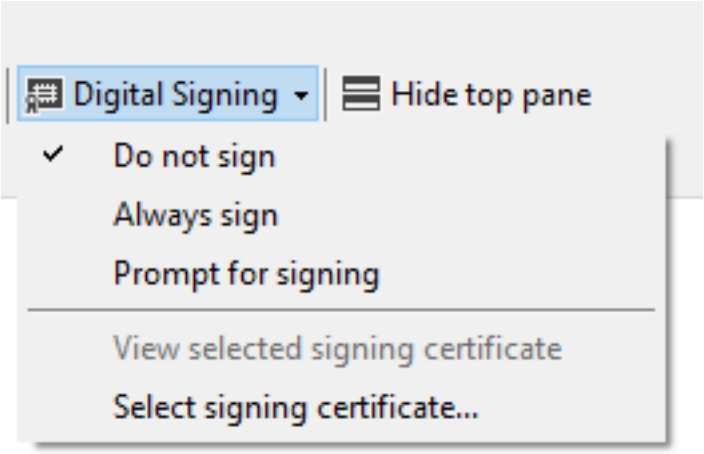
```
<Workflow><![CDATA[{"StateMachine":"State with Approval"}]]></Workflow>
```

1.2.7 Digital signing

You can choose whether or not the instruction XML is digitally signed, and if so, whether or not you should be prompted to sign it when saving an instruction definition.

Instructions...

- do not need to be signed if you only want to develop and test instructions locally in TIMS, or share with other developers
- must be signed with an appropriate code signing certificate if you want to import instructions into a Tachyon system and run them, as described in Running instructions in Tachyon for the first time.



The Digital Signing menu allows you to select a signing certificate that has previously been imported into your certificate store.

In order to use any of the signing options you must have an appropriate certificate. If you do not, the following warning dialog appears



Warning

Known issue: The current version of TIMS does not support CNG code signing certificates. You can select a CNG certificate but when you attempt to save an instruction then you will see an error: *Tachyon Instruction Management Studio: Warning. The file <xml file> could not be digitally signed. Invalid provider type specified.*

1.2.8 Templates

Templates allow you to add visualisation capabilities to your instruction. Visualisation allows you to show results graphically in a variety of formats, as opposed to simply showing tabular results.

At present TIMS does not support the manipulation of visualisations, and requires you to copy a template XML directly into the Template panel, which is then saved as part of the instruction definition.

Tip

See Custom Response Visualizations for details of how to create the XML to insert into this panel.

1.3 Running instructions for the first time

A getting started guide to configuring and verifying your system so that you can run customized instructions.

Aside from the Product Packs provided with Tachyon and the ones available from the , Tachyon lets you add to its functionality by defining your own instructions, either by modifying existing ones or creating them from scratch.

This page describes how to prepare your Tachyon system to run your custom instructions; it does not provide a comprehensive reference for all the methods that you can use when defining your instructions, for that please refer to Writing instructions.

The process requires a Tachyon administrator to use the Server installation account to install a copy of the editor on the server, and test the code signing certificate. The certificate thumbprint and your prefix can then be registered with 1E who will update the Tachyon license after which you can run your instructions.

1.3.1 Prefixes, code signing certificates and licensing

If you want to develop your own custom Tachyon instructions, or modify others, then you will need to sign them using your own code signing certificate so that they can be licensed, imported and run in your Tachyon system. You don't need to do this for instructions that are provided with the product or that have been downloaded from the Tachyon Exchange as they've already been code signed and licensed using the Platform and Exchange certificates from 1E.

Ideally all of your Tachyon instruction developers should share a single code signing certificate between them. Each code signing certificate must be registered in your Tachyon license and associated with a particular instruction name prefix. Ideally you would have one prefix to go with one signing certificate that could be used for all your custom instructions. When you have chosen your prefix and have your code signing certificate(s) you then need to send details of these to 1E, who will update your Tachyon license. This will then automatically activate on your Tachyon Server (assuming it has connection to the Internet).

The following points apply to the importing and running of custom Tachyon instructions:

- Tachyon will only allow instructions to be *imported* if they have been signed and the public key of the code-signing certificate exists in the Tachyon Server's Trusted Publishers certificate store.
- Tachyon will only allow instructions to be *run* if their prefix and the thumbprint of their code-signing certificate have been registered in the Tachyon Server's license file (even if instructions have been successfully imported they will be flagged as unlicensed if the license information is not there).

1.3.1.1 Registering a prefix and code signing certificate and updating the Tachyon license

The following steps may seem complex on initial viewing, but the process has been designed to verify that your code signing certificate can be used to sign instructions before you ask for it to be registered and added to the Tachyon License - as this may take several days to complete you don't want to have to wait for registration only to find that you want to use a different prefix or the signing certificate cannot be used. The general outline of the process is as follows:

- Decide what prefix you will use
- Obtain a code signing certificate, and export as a PFX for use on other computers
- Install the certificate on the Tachyon Server
  - Local user Personal store - for use by TIMS (optionally add to the Local computer store if multiple user accounts will use TIMS on the Server)
  - Local computer Trusted Publishers store - for use by the Server
- Install TIMS and confirm it can see the code signing certificate
- Create a test instruction, sign and import it
- Before you can run the test instruction, you need to register your prefix and certificate thumbprint with 1E
- Once registration is complete you can run the test instruction
- Finally you should delete the test instruction to avoid any confusion

1.3.1.2 Prerequisites

Before you start you will need the following:

- Tachyon Server is already installed and licensed in the lab, has been verified and is connected to the Internet
- TIMS installer MSI

1.3.1.3 Assumptions

There are many ways to define the code signing certificate and configure your Tachyon environment. Here we make some basic assumptions about the type of certificate and Tachyon environment to show the end-to-end process simply. If you want more details on other certification options please refer to the online Tachyon SDK documentation.

These steps are recommended for a lab environment with the following:

- A Microsoft CA has been installed
- A code signing template must be issued on the issuing CA. The default **Code Signing** template is sufficient

- All users will sign Tachyon Instructions on the Tachyon Server
- TIMS must be started as local administrator, so all users developing Tachyon Instructions must be AD domain user accounts that are also members of the Administrators local group on the Tachyon Server. This is one of the requirements for the Tachyon Server installation account anyway.

1.3.2 Steps to register a prefix and code signing certificate to obtain a Tachyon license

1.3.2.1 Decide your Instruction Name Prefix

1.
    a. You should choose a prefix that will be used to name all the instructions created by your TIMS developers.

    b. By default, TIMS will use your logged-on account's domain name as your chosen prefix.

1.3.2.2 Obtain a code signing certificate

Tip

Use this step if you and your fellow TIMS developers do not already have a code signing certificate.

Skip this step if you already have a PFX; the PFX is also useful way to share the certificate with each of your developers.

1.
    a. Log on to the Server using the Server installation account.

    b. Start Certificate Manager for Local User (CertMgr.msc) and request (enroll) a new code signing certificate, ensuring you set the option to allow the private key to be exported.

    Warning

    Enrolling on the Server using the Server installer account means only that account will be able to access the certificate on the Server, and perform the remaining steps including signing instructions using TIMS. You will need to export the certificate with private key if you need to use the certificate elsewhere or others need access it.

    This is why you must enable the Export Private Key when you enroll. If you don't then you will have to start the entire process over again to enroll a new certificate and register it in Tachyon.

    See note on Certificate permissions for more details.

    c. Make a backup copy of your certificate by exporting the code signing certificate as a PFX (including private key) and make the private key exportable. When exporting you must specify a password, which you must protect. If you do not specify a password, then the private key will be accessible only to the user accounts, AD groups and computers which you specified.

Certificate permissions

In some cases, the private key may only be accessible to specific users and not accessible to the installer account - in this case you will need to do the first steps (as well as the steps that involve TIMS) on a developer's PC using the developer's account. You may also be required to import into the Local User store instead of the Local Computer store.

When using Microsoft Certificate Manager to request a new certificate then you must request using Local User instead of Local Computer because this is a requirement of Microsoft's code signing template.

To share the certificate you will need to export it and then provide it to each user to import into their Local User store, or into their Local Computer store, where it will be accessible to all users of the computer.

Tip

The default template for enrolling a code signing certificate does not automatically have **Make private key exportable** enabled. When enrolling using the Microsoft wizard you must click on the Properties button and in Certificate Properties select the Private Key tab and enable the option.

1.3.2.3 Import the certificate to the local computer store

Tip

This step ensures the signing certificate is in a Personal store that is accessible by TIMS.

1.

a. Use the PFX to import the certificate into the Local Computer Personal certificate store on the Server, ensure that the option to make the private key exportable is not set.

Note

If you enrolled your own certificate, then exporting a PFX and then importing back into the Local Computer store allows everyone on the Server to access the certificate and use it to sign instructions in TIMS. By not allowing others to export the private key, you prevent them sharing it with others and you can decide who to give your PFX file to.

If you were given a shared PFX and you are unable to import into the Local Computer store, then import to your Local User store. Each user that needs to use TIMS to sign instructions on the Server will also need to import the PFX into their own Local User store on the Server.

b. Start Certificate Manager and locate the certificate. Open the certificate and confirm you can see it "has a private key that corresponds to this certificate" (see screenshot below).

Exporting the Signing Certificate

You can export the signing certificate with or without the private key. Exporting with a private key (as a PFX) is required only when sharing the certificate with other developers on their machines. You should save the export with a password, or ensure it is permissioned.

Your code signing certificate private key is a valuable secret. If an attacker were to get a copy of the certificate and its private key, they could sign their own instruction XML files. Of course, they would still need to compromise the Tachyon server in order to upload them, and the Tachyon Explorer in order to execute them, but nonetheless, you should treat code signing certificates as protected company assets.

The public key is required by the Tachyon Server. If the certificate is not already present in the Local Computer store, then you will need to export the public key (as a CER file) and import on the Tachyon Server.

1E requires only the thumbprint, or a CER file.

1.3.2.4 Copy the certificate to the Trusted Publishers store on the Server

Tip

This step is required to allow you to import instructions into Tachyon that have been signed using the certificate; if you prefer this step can wait until after the next step.

1.

a. Copy the signing certificate from the Personal Store to the Trusted Publishers store.

b. Reset IIS so that the Consumer API can cache the new certificate.

1.3.2.5 Install TIMS onto the Server

Tip

This step confirms you can see the new signing certificate in TIMS. It also means that at least this certificate can be used to sign instructions using TIMS on the server. You may already have installed TIMS.

1.

a. Logon to the Server as the installer account.

b. Install TIMS.

c. Start TIMS using the installer account.

d. Select the signing certificate (**Code Signing → Select signing certificate**).

e. View the selected signing certificate (**Code Signing → View selected signing certificate**).

f. Change to Always sign (**Code Signing → Always sign**).

g. In the Instruction Definition pane, note the default name of the instruction. This will be **<YourDomainName>-NewInstruction-1**.

1.3.2.6 Edit and Sign the HelloWorldTest instruction

Tip

This step confirms you can create a sign an instruction in TIMS. Later, you will import this signed instruction into Tachyon.

1.

a. Download or copy the test instruction XML to a folder on the Server (this instruction is not already signed).

b. Start TIMS using the installer account.

c. Open the test instruction (File → Open → select **XXXX-HelloWorldTest.xml**).

d. Click Run, to show the Message Hello World in the Results pane.

e. In the Instruction Definition pane, change the instruction name so that XXXX is changed to your chosen prefix.

f. Save the updated test instruction (File → Save As) and confirm the XML file name uses your chosen prefix.

g. At the top of the instruction pane, confirm you can see the 'This instruction definition was signed by <your certificate>'.

Note

If you gave your signing certificate a friendly name, you will see that here, otherwise you will see the subject name of the certificate.

Warning

If you are testing multiple prefixes or signing certificates then you must make the instruction unique and recognisable, by changing the Instruction Name and ReadablePayload.

XXXX-HelloWorldTest.xml

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<InstructionDefinition
Author="1E"
Name="XXXX-HelloWorldTest"
ReadablePayload="Get Hello World Test Message"
Description="Reply Hello World"
InstructionType="Question"
InstructionTtlMinutes="10"
ResponseTtlMinutes="10"
Version="1.0"
xmlns="http://schemas.1e.com/Tachyon/InstructionDefinition/1.0">
  <Payload><![CDATA[SELECT "Hello World" AS Message;]]></Payload>
  <SchemaJson><![CDATA[[
  {
    "Name": "Message",
    "Type": "string",
    "Length": 32
  }
]]]></SchemaJson>
</InstructionDefinition>
```

1.3.2.7 Import the HelloWorldTest instruction into

Tip

This step confirms you can import signed instructions. More specifically, that the Tachyon Server is able to see the certificate in the Trusted Publishers store (and the updated Consumer Web.Config file has taken effect).

1.
    a. Using the installer account, start a browser, connect to , and navigate to the **Instruction Sets** admin page.

    b. Locate the XML file in the folder where you saved the instruction, and drag it onto the drop zone of the Instruction Sets admin page.

    c. Confirm the instruction uploaded without error.

    d. Confirm the instruction is not licensed.

1.3.2.8 Get the Thumbprint of the code signing certificate and send the Thumbprint and Prefix details to 1E

Tip

By this point we know the certificate can be used by TIMS, and will allow your own signed instructions to be imported into Tachyon; the remaining steps will update your license with your prefix and code signing thumbprint so that you can run your signed instructions. In the meantime you can carry on using TIMS to develop and test other instructions.

1.
    a. Get the thumbprint of your signing certificate.

        i. Look at the properties of the code signing certificate, Details tab, and find the Thumbprint (see screenshot).

        ii. Copy this to notepad, remove the spaces, and convert to UPPERCASE.

    b. Get your chosen prefix.

    c. Email them to SalesOps@1e.com.

    d. Currently, the licensing process takes around 3 days. You will receive an email notification that your license has been updated.

    Warning

    The application of the license will only work automatically if your Tachyon Server is connected to the Internet, so that it can re-activate its updated license directly with the 1E License server.

1.3.2.9 Confirm your Server License has been updated

Tip

This step confirms the server has had its license automatically updated.

1.

a. After you have received the license email notification.

b. Using the installer account, start a browser, connect to Tachyon Explorer, and navigate to the **License Info** admin page.

Note

The License Info page will auto-refresh every 10 minutes so you may have to wait for the information to update.

c. Check that the new license has been applied, these instructions will work if you have only registered one Prefix with 1E:

i. Under the **Products** heading expand **Item 2** the **Name** will be **TachyonExplorer**.

ii. Under **Item 2** expand **Instructions** then expand **Item 3**.

iii. Confirm that the Thumbprint and Pattern (Prefix) that you sent in the email to 1E have been added to the license.

Note

The License info page (and the LIC file it is based on) uses the term pattern to mean the same thing as prefix.

1.3.2.10 Confirm the HelloWorldTest instruction is licensed

Tip

This step confirms you can run signed instructions. More specifically, that the Tachyon Server license has been updated to include the thumbprint of your signing certificate, and your chosen prefix.

1.

a. Using the installer account, start a browser, connect to , and navigate to the **Instruction Sets** admin page.

b. Locate the instruction you imported earlier and confirm it is now licensed.

c. Optionally navigate to the home page, and run the instruction. Type **Hello** and select the **Hello World** instruction; set coverage to a specific test machine.

1.3.2.11 Delete the HelloWorldTest instruction

Tip

This step avoids issues that may occur if you re-import the same instruction at a later date using the same ReadablePayload but a different prefix and signing certificate; if you did not delete then importing the same instruction with a different name will result in two instructions with the same ReadablePayload, rather than updating the instruction.

1.

a. Connect to and navigate to the Instruction Sets admin page.

b. Delete the instruction you imported earlier.

1.4 Instruction Definition Reference

Tachyon Instruction Definition format and attributes.

Tachyon Instructions - questions and actions - are defined using an XML format. This format includes the **body** of the Instruction (the logic that the Tachyon Agent will run to execute the Instruction), as well as other metadata which determines how the Instruction is described, categorized and how its lifecycle is managed.

Typically, an Instruction Definition is authored using a tool like Tachyon Instruction Management Studio.

This page is a reference for the instruction definition schema.

This page assumes basic familiarity with XML and JSON formats.

1.4.1 Terminology

You should understand the following terms before reading further:

Term	Explanation
Instruction	An Instruction is an "instance" of when an Instruction Definition is used, created when that Instruction Definition is executed. For example, typing a question fragment into Tachyon Explorer technically shows a list of available Instruction Definitions. At the point that a specific question is selected and the user clicks "Ask this question", a new Instruction instance is created based on the Instruction Definition. You can think of Instruction Definitions as being like templates, and Instructions like things being created from that template. Instructions are transitory, in the sense that they have a finite lifetime.
Instruction Resource	An Instruction Resource is a file, such as a PowerShell script, which is embedded into the Instruction Definition. When a Tachyon Agent runs an Instruction based on this Instruction Definition, the Instruction Resource is available to be downloaded and used by the Agent.
Instruction Payload	The payload of an Instruction is a sequence of steps (expressed in the Tachyon Agent Language) which are sent to, and executed by, the Tachyon Agent when that Instruction is issued.
Product Pack	A Product Pack is the name given to a zip file which contains one or more (normally related) Instruction Definition XML files.

Term	Explanation
	Product Pack zip files can be imported directly into Tachyon Explorer, which will import all the contained Instruction Definitions. **Product Pack structure** Product Pack zip files must contain "flat" set of XML files; in other words, the zip file should not contain any sub-folders.
Task Group	A Task Group is a hierarchical structure used to group together related Instruction Definitions, similar to folders on a file-system. Task Groups are displayed in the Tachyon Explorer "Tasks" page.
Digital Signature	A Digital Signature is a way of guaranteeing that a document (e.g. an XML file) was created by a specific author, and that the document has not been modified or tampered with since it was authored.

1.4.2 Instruction Definition format

An Instruction Definition XML file comprises of an XML document header, an *<InstructionDefinition>* parent XML element, and various sub-elements describing specific aspects of the definition.

The sub-elements are outlined below, some of which are described in more detail in later sections.

At a high-level, the XML format is as follows:

Instruction Definition XML structure

```
+-- (XML document header)
+-- InstructionDefinition
    +-- Comments
    +-- Payload
    +-- ParameterJson
    +-- SchemaJson
    +-- AggregationJson
    +-- Workflow
    +-- ResponseTemplateConfiguration
    +-- Resources
    |   +-- Resource
    |       +-- Content
    +-- TaskGroups
    |   +-- TaskGroup
    +-- Signature
```

Note that child elements of the <InstructionDefinition> element can appear in any order.

The individual elements have the following purposes (elements highlighted in blue are links to more detail further down in the page):

Element	Required	Purpose	Database column
InstructionDefinition\Payload	Yes	Contains the Instruction Payload, written in the Tachyon Agent Language, that should be sent to the Tachyon Agent(s), including placeholders for any parameter values which are substituted when the Instruction is issued. *The content of this element is typically marked up as CDATA within the XML (to simplify escaping of reserved XML characters).*	InstructionDefinition.Payload
InstructionDefinition\Comments	No	Contains optional comments about the Instruction Definition provided by the author.	InstructionDefinition.Comments
InstructionDefinition\SchemaJson	Yes	A JSON representation of the data structure which will be created to store Responses from Tachyon Agents when they reply to this Instruction. *The content of this element is typically marked up as CDATA within the XML.*	InstructionDefinition.SchemaJson
InstructionDefinition\AggregationJson	No	A JSON representation of the data structure which will be created to store summarized ("aggregated") Responses from Tachyon Agents, and the grouping and aggregate operations which define how the data should be summarized. *The content of this element is typically marked up as CDATA within the XML.*	InstructionDefinition.AggregationJson
InstructionDefinition\ParameterJson	No	A JSON representation of the parameters for this Instruction Definition. Parameters can be supplied by the user at the point that the Instruction is issued. *The content of this element is typically marked up as CDATA within the XML.*	InstructionDefinition.ParameterJson

Element	Required	Purpose	Database column
InstructionDefinition\Workflow	No	A JSON representation of the Tachyon Workflow state machine, and other metadata, to be used if the Instruction Definition should not follow the corresponding default approval workflow.	InstructionDefinition.Workflow
InstructionDefinition\ResponseTemplateConfiguration	No	A JSON representation of the logic that Tachyon Consumers may use to provide customized rendering of Responses for the Instruction, such as charts and graphs. This topic is described in more depth here: Custom Response Visualizations	InstructionDefinition.ResponseTemplateConfiguration and InstructionDefinition.ResponseTemplateId
InstructionDefinition\Resources	No	A container element for one or more *<Resource>* child elements, which define the content for embedded resources for this Instruction Definition.	N/A
InstructionDefinition\Resources\Resource	Yes	Defines the name of a resource, and contains one or more *<Content>* child elements.	InstructionDefinitionResource table
InstructionDefinition\Resources\Resource\Content	Yes	Defines the properties and metadata about the (platform-specific) content for an individual resource, and the content itself (represented as base-64 encoded binary). *The content of this element is typically marked up as CDATA within the XML.*	InstructionDefinitionResource.Content
InstructionDefinition\TaskGroups	No	A container element for one or more *<TaskGroup>* child elements.	N/A
InstructionDefinition\TaskGroups\TaskGroup	Yes	Defines the path of a Task Group under which this Instruction Definition should be grouped.	TaskGroups and Task

Element	Required	Purpose	Database column
InstructionDefinition\Signature	No	Contains digital signature information for the XML file.	N/A

1.4.3 Digital Signature

As well as the elements described above, an Instruction Definition XML file will typically include a Digital Signature. This signature provides a mechanism to verify the source and integrity of the file.

> **Digital Signatures and Licensing**
>
> Licenses issued by 1E for Tachyon contain restrictions on the Instruction Definitions which can be imported (based on the name of the Instruction Definition, and who has signed it).

Instruction Definition XML files are signed in accordance with the standard defined by the World Wide Web Consortium. The details of the signing mechanism are beyond the scope of this document, but in summary:

- The XML document contains an additional *<Signature>* element underneath the root *<InstructionDefinition>* element
- The *<Signature>* element contains a number of child elements, which include a Digest Value, a Signature Value, and a base-64 encoded representation of the X509 certificate used to sign the document
- Modifying or invalidating the content of the Digital Signature will effectively render the XML file unsigned. If this happens, the Instruction Definition XML file cannot be imported into Tachyon.

The signing certificate is not required to issue instructions. The system will only save the thumbprint in InstructionDefinition.Thumbprint field in the database and will use this thumbprint along with instruction definition name and payload to create a checksum (InstructionDefinition.Checksum field in the database). Checksum is used to prevent instruction definition from being tampered with directly in the database after it was imported and it is verified each time an instruction is issued based on given definition.

The signing certificate is required in the Trusted Publishers certificate store on the server only in order that an Instruction administrator can import instructions.

1.4.4 Sample Instruction Definition XML

Sample Instruction Definition Expand source

```
<?xml version="1.0" encoding="utf-8"?>
<InstructionDefinition xmlns="http://schemas.1e.com/Tachyon/InstructionDefiniti
```

```
on/1.0"
    Name="1E-Explorer-TachyonCore-FindNonBitLockerDrives"
    InstructionType="Question"
    InstructionTtlMinutes="30"
    ResponseTtlMinutes="30"
    ReadablePayload="Which fixed drives don't have Bitlocker enabled?"
    Description="Returns fixed drives which don't have Bitlocker enabled."
    Version="3">
    <Payload>
<![CDATA[
@script = HttpGetFile(URL : "findNonBitlockerDrives.ps1", Size:917, Hash:"e91ab
19be166e029b71d1dba42eac838141fc23afc247a242f29c6a08b0cbcfa");
Scripting.Run( Language: "PowerShell", ExecutionPolicy : "Override", Script : @
script, InterpretAsJson : true );
]]>
    </Payload>
    <SchemaJson>
<![CDATA[
[
    {
        "Name":  "MountPoint",
        "Type":  "String",
        "Length":  16
    },
    {
        "Name":  "CapacityGB",
        "Type":  "String",
        "Length":  64
    },
    {
        "Name":  "VolumeStatus",
        "Type":  "String",
        "Length":  64
    }
]
]]>
    </SchemaJson>
    <ParameterJson>
<![CDATA[]]>
    </ParameterJson>
    <AggregationJson>
<![CDATA[
{
    "Schema":  [
                    {
                        "Name":  "MountPoint",
                        "Type":  "string",
                        "Length":  512
                    },
                    {
```

```
                    "Name":  "Count",
                    "Type":  "int32"
                }
            ],
    "GroupBy":  "MountPoint",
    "Operations":  [
                    {
                        "Name":  "Count",
                        "Type":  "count"
                    }
                ]
}
]]>
    </AggregationJson>
    <ResponseTemplateConfiguration>
<![CDATA[
{
    "Name":  "default",
    "TemplateConfigurations":  [
                                {
                                    "Id":  "driveChart",
                                    "Title":  "MountPoint vs device count",
                                    "Type":  "Pie",
                                    "X":  "MountPoint",
                                    "Y":  "Count",
                                    "PostProcessor":  "mountPointProcessingF
unction",
                                    "Size":  1,
                                    "Row":  1
                                }
                            ],
    "PostProcessors":  [
                            {
                                "Name":  "mountPointProcessingFunction",
                                "Function":  "ProcessSingleSeries('MountPoint',
'Count', '10', 'Devices with this Mount Point', 'false')"
                            }
                    ]
}
]]>
    </ResponseTemplateConfiguration>
    <Resources>
        <Resource Name="FindNonBitlockerDrives">
            <Content FileName="findNonBitlockerDrives.ps1"
                Size="917"
                Hash="e91ab19be166e029b71d1dba42eac838141fc23afc247a242f29c6a08
b0cbcfa"
                Platform="Windows"
                Type="PowerShell">
                <![CDATA[I2ZpbmRzIGZpeGVkIGRyaXZ ... [omitted for brevity] ...
```

```
bXByZXNzDQpleGl0IDA=]]>
            </Content>
        </Resource>
    </Resources>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
            <Reference URI="">
                <Transforms>
                    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <DigestValue>EFanv4wJtb8NkudpydiCW0gI+7M=</DigestValue>
            </Reference>
        </SignedInfo>
        <SignatureValue>
            fGyi0f6v8xi+l0QrARqVHykyjMwHayis4fnQ+o9XoYDldsAI ... [omitted for brevity] ... YQtND4yptDkK3pvzg==
        </SignatureValue>
        <KeyInfo>
            <X509Data>
                <X509Certificate>
                    MIIGejCCBGKgAwIBA ... [omitted for brevity] ... tYAiJcgeRctvzo8G5B
                </X509Certificate>
            </X509Data>
        </KeyInfo>
    </Signature>
</InstructionDefinition>
```

1.4.5 Instruction Definition XML elements

The best way to get an understanding of the Instruction Definition XML format is to look at specific examples and see how the definition corresponds to each facet of the Instruction Definition.

This section describes the XML format in detail. The following section contains an XSD which can be used to check the validity of an individual Instruction Definition file.

XML Schema Versioning

Note that the XML references an XML namespace: http://schemas.1e.com/Tachyon/InstructionDefinition/1.0

The last part of this namespace references a version number (1.0), which corresponds to the version of the XML format used to express the Instruction Definition.

Future changes to the InstructionDefinition XML format will reference higher version numbers of the namespace for compatibility puropses.

Case Sensitivity

Note that XML and JSON formats are case sensitive with respect to names of elements and attributes.

JSON inside XML

Many of the XML elements inside the Instruction Definition format contain embedded JSON.

The contents for these elements are usually marked up as CDATA within the XML to simplify escaping of reserved XML character sequences.

Since JSON content cannot be validated by an XML schema, the XSD supplied below will be unable to detect problems with JSON formatting, however any such problems can be identified at the point the Instruction Definition is imported in Tachyon (either through Tachyon Explorer or the Tachyon Consumer API).

The JSON formats are described in detail below.

1.4.5.1 InstructionDefinition

This is the root element - aside from the XML declaration element - of the document.

This element represents an individual Instruction Definition, and must appear exactly once within the document.

This element has the following attributes:

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
Name	Yes	string	InstructionDefinition.Name	The name of the Instruction Definition, which must be unique. The name should follow the format *companyName-optionalConsumerName-name*, as Tachyon licenses restrict which instructions can be issued based on a naming	*1E-Explorer-OperatingSystemSummary* *1E-NightWatchman-PowerProfileConfiguration* *Acme-Networking-DhcpAllocationPatterns*

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				pattern. Instruction Definitions whose names do not match a licensed naming pattern will be able to be imported into Tachyon, but will not be able to be used without a corresponding license. Instruction names are limited to 256 characters.	*Acme-SoftwareUsageSummary*
InstructionType	Yes	string (enumeration)	InstructionDefinition.InstructionType	The type of instruction, which determines the default approval workflow that will be used when the instruction is issued. Possible values are: • *Question* - use when the instruction will not alter the state of any device on which it runs	*Question* *Action*

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				• *Action* - use when the instruction may modify the state of a device on which it runs.	
Description	Yes	string	InstructionDefinition.Description	A brief description of what the instruction does when it runs. This description is visible in Tachyon Explorer. Where possible, your description should start with a verb in the present tense (see examples).	*Shows the number of TCP connections by recently executed processes.* *Modifies the Finance application configuration to enable HTTPS.* *Counts the number of times applications (per publisher) have been run.*
ReadablePayload	Yes	string	InstructionDefinition.ReadablePayload	The primary text to show for the instruction, normally phrased in the form of a question or an action.	**Question examples** *Which devices have a USB storage*

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				This text is shown throughout Tachyon Explorer, for example when selecting an instruction from the home page. The Readable Payload must contain substitution placeholders for all parameters which have been defined for the instruction (see examples).	*device connected?* *How many connections has each process made in the last %days% days?* *Which versions of %publisher% %product% are installed?* **Action examples** *Remove all expired certificates from Windows certificate store* *Close any browser processes which are running as a local administrator* *Delete any temporary files with a %extension% file extension*

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
InstructionTtlMinutes	Yes	unsigned integer	InstructionDefinition.InstructionttlMinutes	Also known as the **gather period**. This period starts after the instruction has been approved. The number of minutes, by default, for which Instructions based on this Instruction Definition will be open to receive responses from Tachyon Agents. The possible valid range of values is defined in the Global Settings table within the Tachyon master database. The setting is named **DefaultInstructionTtlMinutes**, and uses JSON to represent a pair of minimum and maximum values. The database default for this range is minimum of 10 minutes, maximum of 1440 minutes (i.e. 1 day). In Tachyon Explorer, this	30

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				value can be overridden by the user, as long as it falls within the minimum and the maximum values (described below), or the default minimum/maximum (defined in the Global Settings table) if one or both of these is not specified for this Instruction Definition.	
ResponseTtlMinutes	Yes	unsigned integer	InstructionDefinition.ResponseTtlMinutes	Also known as the **keep period**. This period starts after the gather period has completed. The number of minutes, by default, for which responses to Instructions based on this Instruction Definition will be kept before they are deleted automatically. The possible valid range of values is defined in the Global Settings table within the	240

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				Tachyon master database. The setting is named **DefaultResponseTtlMinutes**, and uses JSON to represent a pair of minimum and maximum values. The database default for this range is minimum of 10 minutes, maximum of 10080 minutes (i.e. 7 days). In Tachyon Explorer, this value can be overridden by the user, as long as it falls within the minimum and the maximum values (described below), or the default minimum/maximum (defined in the Global Settings table) if one or both of these is not specified for this Instruction Definition.	
Version	Yes	string (repres	InstructionDefinition.Version	A number representing the	3

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
		enting numeric value)		version number of this Instruction Definition. This version number should be incremented by the author whenever a change is made to the Instruction Definition. The version number may be a single integer number, or may be dotted-notation (e.g. 1.2), with up to four components. Uploading an Instruction Definition into Tachyon where a previous definition already exists with the same name but with a *higher* version number will result in an error. Uploading an Instruction Definition into Tachyon where a previous definition already exists with the	1.0

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				same name and the *same* version number will cause the definition being uploaded to be discarded.	
MinimumInstructionTtlMinutes	No	unsigned integer	InstructionTtlRange.Minimum inside InstructionDefinition.Restrictions	Specifies the minimum instruction TTL allowed for this instruction (see above). If this field is omitted a default value taken from GlobalSetting table, entry**DefaultInstructionTtlMinutes** "Minimum" is used.	10
MaximumInstructionTtlMinutes	No	unsigned integer	InstructionTtlRange.Maximum inside InstructionDefinition.Restrictions	Specifies the maximum instruction TTL allowed for this instruction (see above). If this field is omitted a default value taken from GlobalSetting table, entry**DefaultInstructionTtlMinutes** "Maximum" is used.	20

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
MinimumResponseTtlMinutes	No	unsigned integer	ResponseTtlRange.Minimum inside InstructionDefinition.Restrictions	Specifies the minimum response TTL allowed for this instruction (see above). If this field is omitted a default value taken from GlobalSetting table, entry**DefaultResponseTtlMinutes**is used.	10
MaximumResponseTtlMinutes	No	unsigned integer	ResponseTtlRange.Maximum inside InstructionDefinition.Restrictions	Specifies the maximum response TTL allowed for this instruction (see above). If this field is omitted a default value taken from GlobalSetting table, entry**DefaultResponseTtlMinutes**is used.	60
Author	No	string	InstructionDefinition.Author	The name or email address of the author who created this Instruction Definition.	

Attribute Name	Required	DataType	Database Mapping	Description	Example value(s)
				Author names are limited to 256 characters.	

1.4.5.2 SchemaJson

This field will contain JSON array of schema objects inside a CDATA field.

Responses returned for an instruction are of tabular format with one or more columns. SchemaJson defines the data structure that will be used to create table to store responses. It is made of a collection of JSON objects each representing a column.

For example, you may wish to create an instruction which counts the number of TCP connections made per process resulting in responses like:

ProcessName	ConnectionCount
iexplore.exe	*12*
svchost.exe	*4*
ccmexec.exe	*1*
outlook.exe	*13*

To store this kind of responses SchemaJson may look like...

```
[
    {
        "Name":  "ProcessName",
        "Type":  "string",
        "Length": "256"
    },
    {
        "Name":  "ConnectionCount",
        "Type":  "Int32"
    }
]
```

and the actual XML element would look like this:

SchemaJson xml element

```
<SchemaJson>
<![CDATA[
[
```

```
    {
        "Name":  "ProcessName",
        "Type":  "string",
        "Length": "256"
    },
    {
        "Name":  "ConnectionCount",
        "Type":  "Int32"
    }
]
]]>
</SchemaJson>
```

Each column definition (JSON array element) has following properties.

Property Name	Required	Description
Name	Yes	Column name.
Type	Yes	Defines the type of data that can be stored. These data types are based on CLR data types. Supported data types are listed in the table below.
Length	No	Applicable only to String data type but not mandatory to have. If none specified, no size restriction is applied.
RenderAs	No	Defines custom value renderer for the column in Tachyon Explorer. For example, if an instruction returns locations it's possible to display maps in Tachyon Explorer provided a renderer is available for that.

1.4.5.3 Supported Data Types

Name	Description
Bool	Possible values are either true or false.
Int32	Can store any integer value between -2,147,483,648 and 2,147,483,647 inclusive.
Int64	Can store any integer value between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 inclusive.
String	Can store any text data. Depending on size requirements optional length can be specified. If no length is specifed, text of any length can be stored.

Name	Description
DataTimeOffset	Can store date/time along with timezone offset.
Guid	Can store a UUID.
CLOB	Can store a large amount of text. Contents of this type will by default be truncated to a fixed length when the responses are retrieved. In order to retrieve full contents of the field another call to the Consumer API will be required. Truncation length is configurable inside GlobalSetting table, entry ClobDefaultReadSize.

1.4.5.3.1 RenderAs

RenderAs is a hint to a Tachyon consumer that gives additional information on how to render given column.

Tachyon Explorer supports two values:

- ipAddress - this column's value will be formatted as an IP address, with subnet, mask and other information
- preformatted - this column's value will be placed inside a <pre> tag in order to preserve formatting.

Please note this field is just a hint for a Tachyon consumer. It has no effect on how the instruction is processed, how responses are gathered or how the data is stored.

It is up to the consumer to implement support for renderas and decide how to interpret the value. TachyonExplorer supports renderas values used by instruction definitions written by 1E but will not support renderas values introduced by 3rd party.

1.4.5.4 AggregationJson

Depending on the type of data returned from an instruction, it may be appropriate to summarize/aggregate the data before it is presented. For example, you may wish to create an instruction which counts the number of TCP connections made per process, but instead of displaying the results per-device, you wish to display the results aggregated across all devices.

In other words, instead of seeing results which look like this in Explorer:

Device Name	Process Name	Connection Count
device1.acme.or	iexplore.exe	12
device1.acme.org	svchost.exe	4
device1.acme.org	ccmexec.exe	1

Device Name	Process Name	Connection Count
device1.acme.org	outlook.exe	13
device2.acme.org	outlook.exe	12
device2.acme.org	svchost.exe	4
device3.acme.org	iexplore.exe	3
device3.acme.org	svchost.exe	5

Or you may prefer to see results like this:

Process Name	Connection Count
iexplore.exe	*15*
svchost.exe	*13*
ccmexec.exe	*1*
outlook.exe	*25*

This can be achieved by defining an “aggregation schema” for the instruction. This is like performing a GROUP BY and an aggregation function(s) (e.g. COUNT, MIN, SUM, etc.) on the data as a whole, rather than on each device’s set of data individually.

In this case, each Tachyon Agent still executes the same instruction and returns the same data, but the Tachyon Server automatically performs aggregation of that data while it is collected.

Aggregation schema is defined in AggregationJson. As the name suggests this is a JSON object. This depends on SchemaJson as aggregation is done on columns defined in SchemaJson.

For example, given the SchemaJson:

Expand source

```
[
    {
        "Name":  "ProcessName",
        "Type":  "string",
        "Length": "256"
    },
    {
        "Name":  "ConnectionCount",
        "Type":  "Int32"
    }
]
```

Or if we want to achieve the result described above we will need aggregation schema like:

Expand source

```
{
   "Schema":[
      {
         "Name":"ProcessName",
         "Type":"String",
         "Length":256
      },
      {
         "Name":"Count",
         "Type":"int32"
      }
   ],
   "GroupBy":"ProcessName",
   "Operations":[
      {
         "Name":"Count",
         "Type":"sum(ConnectionCount)"
      }
   ]
}
```

and the xml element would look like this:

AggregationJson xml element Expand source

```
<AggregationJson>
<![CDATA[
{
   "Schema":[
      {
         "Name":"ProcessName",
         "Type":"String",
         "Length":256
      },
      {
         "Name":"Count",
         "Type":"int32"
      }
   ],
   "GroupBy":"ProcessName",
   "Operations":[
      {
         "Name":"Count",
         "Type":"sum(ConnectionCount)"
      }
   ]
}
]]>
</AggregationJson>
```

From the example we can see aggregation schema has three properties.

Property Name	Required	Description
Schema	Yes	Defines a list of columns that is used to store aggregated data. These column definitions are the same as the column definitions in SchemaJson. This list combines aggregation operation resultant columns as well as columns against which aggregated data is grouped.
GroupBy	Yes	Defines a comma separated list of column names against which aggregated data is summarised. All the columns must be present in SchemaJson.
Operations	Yes	Defines a list of aggregation operations to be performed. Supported operations are count, min, max and sum. Each operation results in a column that needs to be in the aggregation schema.

1.4.5.4.1 Supported Aggregation Operations

Type	Description
count	Returns the number of items in the group. Resultant data type is Int32.
min(column_name)	Returns the minimum value for the column in the group. Column must exist in in SchemaJson. Resultant column data type is derived from the source data type.
max(column_name)	Returns the maximum value for the column in the group. Column must exist in in SchemaJson. Resultant column data type is derived from the source data type.
sum(column_name)	Returns the sum of all values for the column in the group. Column must exist in in SchemaJson. Resultant column data type is derived from the source data type.

Aggregation schema columns have to exist in either regular schema (see above) and in the GroupBy field or in the Operations array inside the AggregationSchema itself.

1.4.5.5 ParameterJson

Tachyon allows users to pass parameters while issuing an instruction. Allowed parameters are defined in ParameterJson. It is made of a collection of JSON objects each representing a parameter.

For example, we want to know what software is installed published by a publisher where publisher is supplied as parameter by the user. The payload looks like:

```
@allInstallations = Software.GetInstallations();
select * from @allInstallations
where Publisher like '%%publisher%%';
```

This payload needs publisher value to be passed as parameter. The parameter declaration may look like:

```
[
   {
      "Name":"Publisher",
      "Pattern":"%publisher%",
      "DataType":"string",
      "ControlType":"freeText",
      "ControlMetadata":"",
      "Placeholder":"enter publisher name",
      "DefaultValue":null,
      "Validation":{
         "Regex":null,
         "MaxLength":256,
         "AllowedValues":null,
         "NumValueRestrictions":null
      }
   }
]
```

Property Name	Required	Description	Example value
Name	Yes	Name of the parameter. Has to be unique.	MachineAddress
Pattern	Yes	Pattern for the parameter. Has to be unique. This pattern will be substituted with a value provided when an instruction is issued based on this definition. Please note literal text substitution will be performed so ***ANY*** text that matches the pattern will be replaced with parameter value. You should therefore make sure your patterns are unique enough so that unwanted text substitution doesn't occur.	%MachineAddr%

Property Name	Required	Description	Example value
		The pattern does not have to start or end with '%' character, or any specific character and is completely free form.	
DataType	Yes	Data type of this parameter. Supported types: • int (fixed point integer) • float (floating point integer) • string (text field) • date • time	string
ControlType	Yes	Type of control for this parameter which determines what should be shown in the Tachyon Explorer UI. • freeText - UI should display a text box where any text will can be entered and will be validated according to validation rules specified in 'Validation' element • valuePicker - UI should display a value picker with values. A list of allowed values can be defined in the Validation AllowedValues property. • customPropertyKey - UI should display a value picker list of custom property key names of the type specified by Control Metadata • customPropertyValue - UI should display a value picker list of custom property values of the type specified by Control Metadata	freeText
Placeholder	No	Placeholder to display in the UI for this parameter	Address of target machine

Property Name	Required	Description	Example value
DefaultValue	No	Default value for this parameter. Default value must be a valid value for the data type of the parameter ie. you can't assign a string default value to a parameter that has int data type.	
Validation	No	Validation rules for this parameter: • Regex - regular expression, in .NET framework regular expression syntax. Valid only for string free text fields • MaxLength - maximum input string length. Valid only for string parameters. • AllowedValues - array of allowed values. Valid only for value picker entry type • NumValueRestrictions - **Not yet supported**	
ControlMetadata	No	Control metadata. Usage depends on Control Type: • freeText - ignored, not required • valuePicker - ignored, not required • customPropertyKey - must contain type of custom property ie. "CoverageTag" as it exists in the database • customPropertyValue - must contain type of custom property ie. "CoverageTag" as it exists in the database	
Source	No	Used only by custom property value parameters and ignored by all other control	

Property Name	Required	Description	Example value
		types. Points to a related parameter that has the custom property key to use to pr value. In the example below, note how the "Source" for the customPropertyValue parameter is the same as the "Name" of the customPropertyKey parameter.	

Example

```
  <ParameterJson><![CDATA[
[
    {
        "Name":  "Tag name",
        "Pattern":  "%tagname%",
        "DataType":  "string",
        "ControlType":  "customPropertyKey",
        "ControlMetadata":  "CoverageTag",
        "Placeholder":  "Tag name",
        "Validation":  {
                           "MaxLength":  "32",
                           "Regex":  "^[a-zA-Z][a-zA-Z0-9_\\.]{0,31}$"
                       }
    },
    {
        "Name":  "Tag value",
        "Pattern":  "%tagvalue%",
        "DataType":  "string",
        "ControlType":  "customPropertyValue",
        "ControlMetadata":  "CoverageTag",
        "Placeholder":  "Tag value",
        "Source":  "Tag name",
        "Validation":  {
                           "MaxLength":  "1024",
                           "Regex":  "^[^\\|']*$"
                       }
    }
]
```

1.4.5.6 Workflow

...

1.4.5.7 ResponseTemplateConfiguration

A JSON representation of the logic that Tachyon Consumers may use to provide customized rendering of Responses for the Instruction, such as charts and graphs.

This topic is described in more depth here:Custom Response Visualizations.

1.4.5.8 Resources

This element will contain a collection of Resource elements, each of which will be a single resource used by the instruction. Resources are usually script files, but they can be any file format.

1.4.5.8.1 Resource

Resource has a single attribute - Name - by which it is referenced by the script. This element has one child element - Content.

1.4.5.8.1.1 Content

This element contains the actual resource along with some metadata that helps to identify and verify it. All of the metadata attributes are mandatory.

The metadata is in the attributes of the tag:

Attribute name	Description	Example value	Database field
FileName	Name of the file	getSystemInfo.ps1	InstructionDefinitionResource.FileName
Size	Size of the file in bytes	2174	InstructionDefinitionResource.Size
Hash	SHA256 hash of the resource	6051b31bb1a0875f654fcbb5cf5238c1bc249e8a98ed7c0eb685c10db80e2b96	InstructionDefinitionResource.SHA256Checksum
Platform	Platform this resource can be used on. Allowed values are: • Win d	Windows	InstructionDefinitionResource.Platform

Attribute name	Description	Example value	Database field
	ows • MacOS • Linux		
Type	Type of resource	Powershell	InstructionDefinitionResource.Type

While the resource itself is inside the CDATA section in the tag in a Base64 encoded form to prevent issues with illegal characters.

Example Resources section

```
<Resources>
    <Resource Name="GetSystemInfo">
        <Content
            FileName="getSystemInfo.ps1"
            Size="2174"
            Hash="6051b31bb1a0875f654fcbb5cf5238c1bc249e8a98ed7c0eb685c10db80e2
b96"
            Platform="Windows"
            Type="PowerShell"><![CDATA[cGFyYW0oDQo...]]></Content>
    </Resource>
</Resources>
```

1.4.5.9 TaskGroups

Inside this element you will find a collection of TaskGroup elements. Instruction can belong to more than one TaskGroup.

1.4.5.9.1 TaskGroup

Each TaskGroup element has a single a attribute - Name - which defines an arbitrary path to a Task group underneath which this instruction should appear.

If that path does not exist it is created (the TaskGroup entries are created),

Example TaskGroups

```
<TaskGroups>
    <TaskGroup Name="Software Vendors\Microsoft\Configuration Manager" />
</TaskGroups>
```

1.4.6 XSD schema

The following XSD can be used to validation Instruction Definition XML files.

XSD schema Expand source

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           targetNamespace="http://schemas.1e.com/Tachyon/InstructionDefinition
/1.0"
           xmlns="http://schemas.1e.com/Tachyon/InstructionDefinition/1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="InstructionDefinition">
    <xs:complexType>
      <xs:all>
        <xs:element name="Payload" type="xs:string" minOccurs="1" maxOccurs="1"
/>
        <xs:element name="SchemaJson" type="xs:string" minOccurs="1" maxOccurs=
"1" />
        <xs:element name="AggregationJson" type="xs:string" minOccurs="0" maxOc
curs="1" />
        <xs:element name="ParameterJson" type="xs:string" minOccurs="0" maxOccu
rs="1" />
        <xs:element name="Workflow" type="xs:string" minOccurs="0" maxOccurs="1
" />
        <xs:element name="ResponseTemplateConfiguration" minOccurs="0" maxOccur
s="1" />
        <xs:element name="Comments" minOccurs="0" maxOccurs="1" />
        <xs:element name="Resources" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Resource" type="resourceType" minOccurs="1" max
Occurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="TaskGroups" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TaskGroup" type="taskGroupType" minOccurs="1" m
axOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

```
      </xs:all>
      <xs:attribute name="Name" type="xs:string" use="required" />
      <xs:attribute name="InstructionType" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Question" />
            <xs:enumeration value="Action" />
            <xs:enumeration value="EventSubscription" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="Description" type="xs:string" use="required" />
      <xs:attribute name="ReadablePayload" type="xs:string" use="required" />
      <xs:attribute name="InstructionTtlMinutes" type="xs:unsignedInt" use="req
uired" />
      <xs:attribute name="ResponseTtlMinutes" type="xs:unsignedInt" use="requir
ed" />
      <xs:attribute name="Version" type="xs:string" use="required" />
      <!-- The following attributes end up the in 'Restrictions' field on Instr
uctionDefinition -->
      <xs:attribute name="MinimumInstructionTtlMinutes" type="xs:unsignedInt" u
se="optional" />
      <xs:attribute name="MaximumInstructionTtlMinutes" type="xs:unsignedInt" u
se="optional" />
      <xs:attribute name="MinimumResponseTtlMinutes" type="xs:unsignedInt" use=
"optional" />
      <xs:attribute name="MaximumResponseTtlMinutes" type="xs:unsignedInt" use=
"optional" />
      <xs:attribute name="Author" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:complexType name="resourceType">
    <xs:sequence>
      <xs:element name="Content" type="contentType" minOccurs="1" maxOccurs="un
bounded" />
    </xs:sequence>
    <xs:attribute name="Name" use="required"/>
  </xs:complexType>
  <xs:complexType name="contentType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Type" type="xs:string" use="required" />
        <xs:attribute name="FileName" type="xs:string" use="required" />
        <xs:attribute name="Size" type="xs:unsignedLong" use="required" />
        <xs:attribute name="Hash" type="xs:string" use="required" />
        <xs:attribute name="Platform" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Windows" />
                            <xs:enumeration value="MacOS" />
```

```
            <xs:enumeration value="Linux" />
                        </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
    <!-- Can use this online tool to generate hash - https://md5file.com/calcul
ator -->
  </xs:complexType>
  <xs:complexType name="taskGroupType">
    <xs:attribute name="Name" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```

1.5 SCALE User Guide

Understanding the Tachyon client language: SCALE - Simple Cross-platform Agent Language for Extensibility.

The Tachyon Agent features a simple but powerful interpreted programming language, and it is from this language that Tachyon instructions (questions and actions) are built. The language is cross-platform, allowing the same instruction to run on different operating systems, and has support for SQL SELECT commands to allow data to be transformed, filtered and aggregated. The language is concise and easy to learn, and mastery of the language is the key to developing powerful instructions. Version information is provided to identify new features that are not available in older versions of Tachyon.

1.5.1 Quick Start

If you're already familiar with scripting languages and/or SQL, you'll more than likely be able to get started just by skim-reading this document and looking at some of the examples. If that's the case, here's a quick summary of the language which should help:

- The language is interpreted, and mixes SQL SELECT statements with "method" calls on "modules" found in the Functions and Methods Reference.
- The SQL SELECT support is provided by an enhanced version of the SQLite engine.
- Additional custom SQL functions and Agent functions are described below.
- The language is *case sensitive* (although case sensitivity is more relaxed within SQL SELECT statements) and is not whitespace sensitive.
- Statements in the language must be terminated by a semi-colon (;)The Tachyon Agent takes advantage of SQLite extensibility by providing a further "application defined functions" that are built-in to the Agent and can also be used in SELECT statements
- Variables within the language are written as "@myTableName" they are always tables of data that can have one or more rows and one or more columns.
- The output of method calls and SQL SELECTs can be assigned to @tables and can be SELECTed from

- The output of the instruction is always the last set of data produced by a statement, that is the final SELECT statement or method call.
- If a statement produces an error, execution of the instruction terminates immediately with an Exit Code. Any changes made by the Agent are not rolled back.
- There is some simple flow control with FOREACH looping (new in v3.0) and IF/ELSE/ENDIF conditional branching (new in v3.3).
- You can use the "EVALUATE" statement to break out of an instruction at any point if the last statement produced an empty set of data; it is in effect "continue if data available".
- RETURN, NOCONTENT, ERROR, and NOTIMPLEMENTED (new in v3.3) allows Early exit termination of the instruction with different Exit Codes.
- Use double-quotes (") to enclose string literals, and in strings use the sequences backslash-double-quote (\") and backslash-backslash (\\) to escape literal double-quotes (") and a literal backslashes (\) respectively.
- Use C-style comments in Agent language. Both the block and single-line; e.g. */* This is a comment until the block is closed */* and *// This is a comment until the end of the line*.
- Use SQL-style comments in SQL statements. Both the block and single-line; e.g. */* This is a comment until the block is closed */* and *-- This is a comment until the end of the line*.

Here's an example demonstrating some of the key features of the language – it brings back a row for each loaded DLL/EXE by any running process, the file version info data for that binary, and its MD5 hash. Since the example uses WMI, it will run on Windows only.

Example - query file version info and MD5 hash for loaded binaries

```
/* Get all the loaded DLLs and EXEs using a WMI query */
@binaries = NativeServices.RunWmiQuery(Namespace: "root\\cimv2", Query: "SELECT
* FROM CIM_ProcessExecutable");

/* Extract the filename and count the number of times its loaded */
@binaryNames = SELECT   COUNT(1) AS LoadedCount
               ,        LOWER(REPLACE(REGEXPREPLACE("(.*)Name=\"(.*)\"", Antece
dent, "$2"), "\\\\", "\\")) AS FileName
               FROM     @binaries
               GROUP BY FileName;

/* For each loaded binary, get the file version info */
@binaryInfo = FOREACH @r IN @binaryNames DO
  FileSystem.GetVersionInfo(FileName: @r.FileName);
DONE;

/* Then, for each loaded binary, get the MD5 hash */
@hashes = FOREACH @r IN @binaryNames DO
  FileSystem.GetHash(FileName: @r.FileName, Algorithm: "MD5");
DONE;

/* Then bring everything together */
```

```
SELECT     I.*
,          H.FileHash
,          N.LoadedCount
FROM       @binaryInfo AS I
INNER JOIN @hashes AS H      ON I.FileName = H.FileName
INNER JOIN @binaryNames AS N ON I.FileName = N.FileName;
```

1.5.2 Key Concepts

1.5.2.1 Modules and Methods

The Tachyon Agent ships with a number of *modules*, and each module has a number of *methods*. Modules are simply a container, and provide a way of logically grouping methods together. Methods are operations which interact with the device and provide output data in the form of a table. A method may also take one or more *parameters* (see section below). If you have used any kind of object-oriented programming language before, these concepts should be familiar.

An example of a module may be the "Network" module; a corresponding method on that module may be "GetConnections". Calling this method returns a table of data which contains a row for each network connection, with corresponding columns for "IpAddress", "Port", etc. Some methods return only a single value.

The full list of methods is too long to include here, but can be found in the Tachyon Agent Methods reference.

In addition, the Agent language includes some cross-platform built-in functions – think of them like methods available in a "global" module. See Tachyon Agent Built-in Functions reference.

The Tachyon Agent's architecture allows new modules (and therefore new capabilities) to be deployed after the Agent has been installed.

1.5.2.2 Method Parameters

An individual method may take one or more parameters which will determine the behaviour of that method. For example, the "Users" module has a method named "GetLocalGroupMembers". This method takes a parameter named "GroupName" which is the name of the local group whose members should be queried.

A parameter may be mandatory, in which case you must provide a value for that parameter when calling the method, or optional, in which case it can be omitted.

A parameter has a corresponding data type, which determines the possible values that can be supplied for that parameter. For example, a parameter may be a "string", or an "integer", etc.

Method parameters should not be confused with instruction parameters. However, you can use instruction parameters as method parameters.

Method names and their parameter names are **case-sensitive**.

1.5.2.3 Method Output

The output of a method is always a table of data.

The output can be assigned to a @table. Once data is in a @table, it can be processed using SQL, iterated over using FOREACH, and used as a parameter for subsequent method calls.

These concepts are discussed below.

1.5.2.4 SQL

The Tachyon Agent language integrates with SQLite – an open source, light-weight database engine with comprehensive support for the SQL querying syntax. SQLite is a popular, high-performance, feature-rich and light-weight database technology which is incredibly pervasive in modern IT – indeed, it is speculated that SQLite is one of, if not the most widely deployed software component in the world.

Although SQLite has full SQL support, including advanced concepts like views, triggers, pragmas, etc., the Agent Language supports only SQL SELECT statements. Statements such as UPDATE, INSERT and DELETE are not supported within the language. However, by using SQLite to its fullest, it is possible to perform very complex data processing. @tables within the Agent language manifest themselves as temporary tables which can be queried using SQL SELECT statements.

Documentation for SQLite is found on its website. There are other help sites and forums you can search for help and examples. The following are some useful pages:

- https://sqlite.org/lang_select.html - SELECT syntax
- https://sqlite.org/lang_expr.html - includes details of SQL operators
- https://sqlite.org/lang_corefunc.html - SQL core functions
- https://sqlite.org/lang_aggfunc.html
- https://sqlite.org/lang_datefunc.html

SQL is only used to process @tables that are used as parameters and output of Methods. It is not possible to call Methods from within a SQL SELECT statement, therefore if you require some form of flow control then use the Agent Language FOREACH loop. Nor can you use a SQL SELECT statement as a Method parameter; instead use SQL to save a parameter as a @table and use that @table in the Method.

In addition to SQLite functions, the Agent language takes advantage of SQLite extensibility by providing "application defined functions" for use in SELECT statements. See Tachyon Agent SQL Functions reference.

Convention

We suggest you write all SQL keywords in UPPERCASE.

1.5.3 Basic syntax

1.5.3.1 Simple method execution

A Tachyon instruction is composed of one or more sequential statements in the Agent language, separated by a semi-colon. At its simplest, this might be a single method invocation:

Example - query all active network connections

```
Network.GetConnections();
```

This example executes the GetConnections method on the Network module.

The set of data returned from this instruction might look as follows:

Class	LocalAddress	LocalPort	ProcessId	ProcessName	RemoteAddress	RemotePort
IPv4	192.168.11.180	1028	17808	explorer.exe	40.77.229.43	443
IPv4	192.168.11.180	1055	4	(system)	192.168.2.119	445
IPv4	192.168.11.180	1193	32384	lync.exe	192.168.3.152	5061
IPv4	192.168.11.180	1750	1096	Tachyon.Agent.exe	192.168.2.124	4000
IPv4	192.168.11.180	6753	1408	svchost.exe	192.168.30.217	49401
IPv4	192.168.11.180	9262	25536	OUTLOOK.EXE	40.101.9.178	443
IPv4	192.168.11.180	9312	2628	spoolsv.exe	192.168.2.37	49163
IPv4	192.168.11.180	10116	20060	CcmExec.exe	192.168.2.43	10123
IPv4	192.168.11.180	10843	20824	chrome.exe	64.233.166.188	5228
IPv4	192.168.11.180	17108	25536	OUTLOOK.EXE	40.101.72.194	443
IPv4	192.168.11.180	17117	25536	OUTLOOK.EXE	40.101.72.194	443
IPv4	192.168.11.180	17194	25536	OUTLOOK.EXE	40.101.72.194	443
IPv4	192.168.11.180	17201	25536	OUTLOOK.EXE	40.101.72.194	443
IPv4	192.168.11.180	17202	25536	OUTLOOK.EXE	40.101.72.194	443
IPv4	192.168.11.180	17240	25536	OUTLOOK.EXE	40.101.54.2	443
IPv4	192.168.11.180	17242	25536	OUTLOOK.EXE	40.101.54.2	443

Implicit in this statement is the "return" of the output data from this method call as the result of the instruction. Each Tachyon instruction returns a single set of data (in the form of a table), and it is this data that is sent back to the Tachyon Server and ultimately displayed in Tachyon Explorer. The data returned from an instruction is based on the *last set of data the instruction produces.*

For example, the following instruction:

Example - query network connections, then logged on users

```
Network.GetConnections();
Users.GetLoggedOnUsers();
```

… will retrieve the active network connections, but then immediately discard the output. It will then query the logged-on users, and return that output as the result of the instruction (as that is the set of data produced).

The output of this instruction might look as follows:

Account	LogonTime	RemoteIp	RemoteName	RemoteSession	SessionId	SessionName	SessionState
1E\james.davies	07/04/2017 09:00			False	5	Console	Active

Although the Agent language permits this type of construct, it is clearly not useful in this context – two methods are executed, but only the data from the last is available. Things start to come together when the data from multiple method executions are combined using SQL. This is discussed in detail later.

Use of white-space

Although this example shows the two method statements on separate lines for readability, whitespace within the language is not treated as significant. The two statements could appear on the same line.

Case sensitivity

The Agent Language is case-sensitive with respect to names of modules, methods and parameters.

Although some aspects of the Agent Language are more permissive in terms of character case, it is recommended that you treat the entire language as case sensitive.

1.5.3.2 Strings and Escaping

String literals should be enclosed in double-quotes (").

> **Warning**
>
> Single-quotes (') are deprecated. Despite being permitted in SQLite, they should not be used. They are acceptable in the current version of Tachyon but this may change in the future due to potential difficulties with escaping.

The escape character is a backslash (\).

If a literal double-quote is required in a string then use the escape character (\").

If a literal escape character is required in a string then use backslash-backslash (\\).

The following is an example of both uses.

Example - String escaping

```
// Result: The Tachyon Agent log is found in "C:\ProgramData\1E\Tachyon\" by de
fault.

SELECT "The Tachyon Agent log is found in \"C:\\ProgramData\\1E\\Tachyon\\\" by
default." AS Tip;
```

The following examples are different ways of achieving the same thing, depending on your preference for readability.

Example - String concatenation

```
// Result: James says "I would like sausages today" every morning.

@breakfast = SELECT "sausages" AS choice;

SELECT "James says \"I would like " || choice || " today\" every morning." AS g
reeting FROM @breakfast;

SELECT "James says " || CHAR(34) || "I would like " || choice || " today" || CH
AR(34) || " every morning." AS greeting FROM @breakfast;

SELECT PRINTF("James says \"I would like %s today\" every morning.", choice) AS
greeting FROM @breakfast;

SELECT PRINTF("James says %sI would like %s today%s every morning.", CHAR(34),
choice, "\"") AS greeting FROM @breakfast;
```

1.5.3.3 Passing method parameters

Now we'll consider passing parameters to a method. Parameters are passed by specifying their name and value within the parentheses following a method name. Some examples:

Example - passing simple parameters

```
/* Pass a string parameter */
Users.GetLocalGroupMembers(GroupName: "Administrators");
/* Pass an integer parameter */
Agent.Sleep(Seconds: 5);
/* Pass a string parameter and a boolean parameter */
FileSystem.GetFilesInFolder(Folder: "D:\\MyData\\MyPictures", Recursive: true);
```

Note the following about parameters:

- Parameter names are case-sensitive
- Parameters can be supplied in any order
- Parameters can be omitted if they are optional
- Parameter values have datatypes – strings are enclosed in double-quotes (and use the backslash character to escape either a literal double quote or a backslash itself); numbers and Booleans should not be enclosed in double-quotes
  - For Tachyon Agent versions 3.2 and onwards, a boolean can also be supplied as an sqlite3 `integer`. In such a case, zero will be interpreted as `false`, and any non-zero integer will be interpreted as `true`. A `real` will not be accepted. For example:

    v3.3+

    ```
    // Recursion on
    FileSystem.GetFilesInFolder(Folder:"c:\\potato", Recursive: true)
    FileSystem.GetFilesInFolder(Folder:"c:\\potato", Recursive: -15)
    FileSystem.GetFilesInFolder(Folder:"c:\\potato", Recursive: 65535)
    // Recursion off
    FileSystem.GetFilesInFolder(Folder:"c:\\potato", Recursive: false)
    FileSystem.GetFilesInFolder(Folder:"c:\\potato", Recursive: 0)
    ```

- Parameter values can be supplied using @table – this is an advanced topic discussed in Passing @tables as method parameters

1.5.3.4 @tables

@tables have the following properties:

- they are used to store the output of Agent methods and SQL queries, and must be considered as SQLite tables, and as such can have multiple rows and columns, as well as empty tables with no rows.
- their names must be alphanumeric, and start with a letter, and are case-sensitive
- they can be used as parameters for Agent methods
- they can be re-created and re-used multiple times within an instruction

Example - @tables

```
@device = Device.GetSummary();
@device = SELECT Manufacturer, Model FROM @device;
```

You can also store a numeric or string value, without the need for a full SELECT statement. The column name is Value.

Example - @tables

```
@table = 99;
SELECT Value FROM @table;
```

You can create your own @tables. One method is:

```
@colors =
  SELECT
    column1 AS ColorName, column2 AS ColorCode
  FROM
  ( VALUES
    ("Green","#008000"),
    ("Blue", "#0000FF"),
    ("Yellow", "#FFFF00"),
    ("Cyan", "#00FFFF"),
    ("Red", "#FF0000"),
    ("Purple", "#800080")
   );
```

@tables are deleted and memory freed when an instruction exits.

SQLite tables have a built-in autoincremented key **rowid** which is not included in output, unless specified in a SELECT statement. An example of where **rowid** is useful is when using SplitLines in methods, for example in NativeServices.RunCommand and Utilities.SplitLines.

The Tachyon built-in function IsUndefined is used to determine whether a table is real and full of values or whether it is the result of an operation that produced nothing, which is different to producing an empty table.

1.5.3.5 SQL queries

Having executed a method, the data can be returned directly from the instruction, or can be assigned to a @table for further processing.

In the following example, we capture all the active network connections, assign them to a @table, and then run a SQL SELECT query to count them, grouped by the process name (i.e. count number of network connections per process):

Example - count network connections by process

```
@connections = Network.GetConnections();

@connections = SELECT   COUNT(1) AS ConnectionCount
,        ProcessName
FROM     @connections
GROUP BY ProcessName;
```

Since the last statement of this instruction is our SQL query, the output of this query becomes the result of the instruction. The example shows the results of the SQL query are saved in a table called @connections which could be used by further methods or queries.

See SQL section above for useful references on the SQLite website.

1.5.3.6 Reserved keywords

The following are keywords, which are reserved, and should not be used as names of @tables or columns. Potential future keywords should also be avoided because any instructions that use these words may be invalidated, so do not use them.)

1.0	v3.0.0	v3.3	Potential future
• *EVALUATE*	• *DO*	• *IF*	• *ANY*
• *FALSE*	• *DONE*	• *ELSE*	• *BREAK*
• *TRUE*	• *FOREACH*	• *ENDIF*	• *EMPTY*
	• *IN*	• *ERROR*	• *NONE*
		• *NOCONTENT*	• *THEN*
		• *NOTIMPLEMENTED*	• *WHILE*
		• *RETURN*	

1.5.3.7 Comments

Comments can either be multi-line or single-line.

Outside of SQL statements the Agent language supports C-style comments.

Agent language examples

```
/* This is a comment
   which spans
   multiple lines */

@connections = Network.GetConnections(); // This comment continues until the ne
xt line break
```

Within SQL statements, comments are are subject to what is supported by SQLite (see http://www.sqlite.org/lang_comment.html).

SQL examples

```
/* This is a comment
   which spans
   multiple lines */

@flag = SELECT 1 AS value; -- This comment continues until the next line break
```

Multi-line comments are supported in all cases, and can also be used for single-line, provided you remember to terminate the comment.

General example

```
@flag = SELECT 1 AS value; /* This is a comment that has been correctly termina
ted */
```

Known issue

Comments within SQL statements should not include an unterminated quote, single or double. This is not a problem outside of a SQL statement.

Use of comments

Although comments are useful to annotate code, they do not serve any functional purpose. As such, you may wish to consider removing comments from your instructions, otherwise you will be sending comments to each Agent over the network only for them to be ignored when they are processed. At large scales, making small savings on payload sizes (e.g. by removing comments or reducing whitespace) can help to reduce network traffic.

1.5.4 From Agent Language to Tachyon Instruction

1.5.4.1 Writing and testing Instructions

Creating an instruction using the Agent Language is the first step in building a question or an action that can be used from Tachyon Explorer (or any other Tachyon consumer).

The easiest way to create an instruction is to refine and test it on a single device, and then incorporate it into a Product Pack which can be imported into Tachyon. 1E provides the Tachyon Instruction Management Studio (TIMS) which can be used to create and run instructions on your local machine. This tool also allows you to add your instruction into a Product Pack.

See Getting started with TIMS for how to use TIMS.

Continue reading to understand the Agent language.

Use the Functions and Methods Reference to use the power of Tachyon.

1.5.4.2 Instructions and Instruction Sets

Once the instruction has been written and tested, the instruction is then saved as an Instruction Definition, which is simply an XML file.

The definition of an instruction is explained fully in Instruction Definition Reference, but here is a summary:

- A unique name
- The type of instruction – whether it is a question or an action (which affects how it is displayed in Tachyon Explorer). Note that the Agent itself does not differentiate between questions and actions – it considers both as simply "instructions".
- A "readable payload" – this is the instruction phrased as a question or action, and is what is displayed in Tachyon Explorer
- Description - this is additional information that can also be displayed in the Tachyon Explorer; along with the Readable Payload, the Description is also parsed when searching in Tachyon Explorer

- The instruction "payload" – this is the Agent Language script which will be sent to the target devices
- Parameters for the instruction – this allows anyone running the instruction to substitute custom values within the payload
- The instruction "schema" – this describes the structure of the data which the instruction returns, and is used to create the dynamic storage for the results of the instruction on the Tachyon Server
- The instruction "aggregation schema" – this describes how data from multiple devices should be summarized, and is conceptually similar to a "GROUP BY" clause in SQL
- The time-to-live (TTL) values for the instruction – how long to gather data (**gather period - InstructionTtlMinutes**) and how long to keep it (**keep period - ResponseTtlMinutes**)

A Product Pack can also include "resources", such as script files (e.g. PowerShell, BASH, etc.). Adding resources to a Product Pack makes them available for download by the Tachyon Agents. Using scripts within the Agent Language is described in detail later.

1.5.4.3 Instruction Parameters

Adding parameters to an instruction can make it flexible and enables the instruction to be reused. An instruction parameter is simply a substitution of a user-supplied value into the instruction's payload.

In the following example, we extend the previous example by counting network connections per process where the connection is on a specific, user-supplied TCP port.

Example - parameter placeholders

```
@connections = Network.GetConnections();

SELECT   COUNT(1) AS ConnectionCount
,        ProcessName
FROM     @connections
WHERE    RemotePort = CAST("%MyPort%" AS NUMBER)
GROUP BY ProcessName;
```

We would then define a parameter to substitute the placeholder %MyPort% with a value supplied by the user. For guidance on using TIMS to define instruction parameters, please refer to Getting started with TIMS - Instruction Parameters.

Instruction parameter considerations

Since parameter values are substituted directly into the Agent's payload, you must take care to avoid the possibility of SQL or statement injection. Tachyon takes precautions against this (e.g. escaping double-quotes and backslashes within string parameter values), but you should consider the following when parameterizing instructions:

- Although the format %ParameterName% is the preferred convention for a parameter placeholder, you are free to use any substitution placeholder you wish
- Parameters should be defined as the most appropriate data type (string, integer, float, Boolean)

- String parameters should, where possible, be constrained using a regular expression to prevent invalid input
- Placeholders for string parameters should be placed within double-quotes within the Agent Language – the Tachyon Consumer API will automatically perform escaping of the parameter's value
- Placeholders for non-string parameters should not be placed in double-quotes
- Within SQL statements, you may wish to consider using a CAST("%parameter%" AS NUMBER) expression for type-safety
- Although it is possible to use parameters to build "dynamic statements" within the Agent Language, it is NOT recommended unless you take extreme care (and limit the possible values which can be supplied for the parameter)

1.5.4.4 Handling multiple values in a single string instruction parameter

```
/* Transpose a comma-delimited string of KB numbers into a column, eliminating whitespace and blanks */
@kblist = Utilities.SplitLines(Text:"%kblist%", Delimiter:",");
@kblist = SELECT DISTINCT TRIM(Output) AS kb FROM @kblist WHERE TRIM(Output) <> "" ORDER BY kb;
```

1.5.4.5 Output content and column names

Column naming rules

- only use basic alphanumeric characters when naming a column (A-Z, a-z, 0-9) - do not include spaces or other characters
- must start with a letter - so "*MyColumn*", "*Value1*" and "*Iteration3Data*" are valid column names, whereas "*my_column*", "*my column*", and "*3IterationData*" are not
- must be unique when considered case-insensitively - for example, you should not have instructions which returns columns named, "*TEST*", "*test*" and "*Test*", as this will cause conflicts.

The names of the columns within the data set returned by the Agent Language correspond to the column headings for the data when displayed in Tachyon Explorer. Tachyon Explorer performs a basic translation of the name based on changes in upper/lower-case. For example, **ThisIsMyColumnName** will be displayed as **This Is My Column Name**.

You can use **column aliasing** within SQL to name (or rename) a column. For example the instruction below returns two columns of data, one named **ConnectionCount** and the other named **NameOfTheProcess**. Tachyon Explorer will display these as **Connection Count** and **Name Of The Process** respectively.

Example - providing an alias for a column

```
@connections = Network.GetConnections();
SELECT   COUNT(1) AS ConnectionCount
,        ProcessName AS NameOfTheProcess
FROM     @connections
GROUP BY NameOfTheProcess;
```

You also have an alternative option for changing Column names displayed in Explorer by using the **Render As** option when defining the output schema of an instruction, as described in Getting started with TIMS: Adding a schema.

1.5.4.6 Aggregating data

We mention this here, just so you do not confuse aggregating data using SQLite with response aggregation at the server. The latter is not part of the Agent language, but is determined by the AggregationJson specified in the instruction definition. Then the Explorer portal will display the aggregated responses with a drill down into the detailed data.

If you don't need the detailed data, then aggregating data on each device is useful to reduce the quantity of data being reported back to the server and having to be processed.

Example - querying historic captured data

```
@products = Software.GetInstallations();
SELECT   Publisher
,        Count(*) AS Products
FROM     @products
GROUP BY Publisher;
```

1.5.5 SELECTing Data

Once you have data in a @table (e.g. by capturing the output of a method invocation), that @table is available as a pseudo-table within SQLite, which can then be queried.

A @table assignment is straight-forward:

Example - @table assignment

```
@MyTable = Network.GetConnections();
SELECT DISTINCT ProcessName FROM @MyTable;
```

Once created, a @table has "scope" for the entire instruction – in other words, it will exist until the instruction completes, and can be referenced as many times as required within the remainder of the instruction.

@tables can be re-assigned if required, for example:

Example - @table reassignment

```
@MyTable = Network.GetConnections();
@MyTable = SELECT DISTINCT ProcessName FROM @MyTable;
```

In the example above, we assign @MyTable first to the output of querying for network connections, and to the result of SELECTing the DISTINCT (unique) process names from the previous data.

1.5.5.1 Limiting Data

From v3.2, the maximum number of output rows from a single *SELECT_expression* is limited to the value specified by the ***SelectRowsLimit*** configuration value. The range is 1 to 1000000000

(10^9), with a default of 100000. This is intended to prevent runaway expressions that generate a ridiculous number of rows, consuming gigabytes of memory and wasting CPU and time. No further rows are produced if this value is exceeded, but it is *not* considered to be an error situation although a warning is generated in the log file.

1.5.5.2 JOINing and GROUPing

A SELECT statement can of course reference more than one @table if required. The most common reason for doing this is to join data together; for example:

Example - JOINing and GROUPing

```
@p = OperatingSystem.GetProcesses();
@c = Network.GetConnections();

SELECT     COUNT(1) AS ConnectionCount
,          @p.AccountName
,          @c.RemotePort
FROM       @p
INNER JOIN @c
  ON       @p.ProcessId = @c.ProcessId
GROUP BY   @p.AccountName
,          @c.RemotePort;
```

This simple example returns a count of active TCP connections per-port, per-user, by performing an INNER JOIN on the process and connection data using the common "ProcessId" field.

1.5.5.3 SELECTing "out of thin air"

Example - SELECTing without a table

```
/* Note that omitting the FROM clause means we SELECT from a non-existent
   table which has exactly one row. This is similar to using the "dual"
   table in Oracle. */
@d = SELECT DATETIME() AS LocalDate;
```

1.5.5.4 More complex examples

SQLite provides huge flexibility and power in its SELECT syntax – the full syntax is available online.

Here are some more examples to get you started.

1.5.5.4.1 Get the top 5 processes with the most number of active TCP connections

Example - using LIMIT

```
@c = Network.GetConnections();

SELECT   COUNT(1) AS ConnectionCount
,        ProcessName
FROM     @c
GROUP BY ProcessName
```

```
ORDER BY ConnectionCount DESC
LIMIT    5; /* Use LIMIT n to return the first n rows */
```

1.5.5.4.2 String concatenation

Example - basic string concatenation

```
@p = OperatingSystem.GetProcesses();

SELECT Executable || " was run by " || AccountName AS Message
FROM @p;
```

Concatenation from multiple @tables:

Example - string concatenation

```
@programData = NativeServices.RunCommand(CommandLine:"cmd /c echo %PROGRAMDATA%
");
@tachyonFolder = SELECT "\\1E\\Tachyon" AS Path;

SELECT @programData.Output || @tachyonFolder.Path AS TachyonPath FROM @programD
ata,@tachyonFolder
```

1.5.5.4.3 Using CASE to translate values

Example - SELECT CASE

```
@c = Network.GetConnections();

SELECT CASE
 WHEN RemotePort = 443  THEN  "HTTPS"
 WHEN RemotePort = 80   THEN  "HTTP"
 WHEN RemotePort = 21   THEN  "FTP"
 WHEN RemotePort = 4000 THEN  "1E Tachyon"
 ELSE "Port #" || CAST(RemotePort AS STRING)
 END AS PortName,
 ProcessName,
 RemoteAddress
FROM @c;
```

1.5.5.4.4 Sub-SELECTs

Example - Sub-SELECT

```
@bit = NativeServices.RunCommand(CommandLine:"manage-bde -status", SplitLines:t
rue);
@baseLN = SELECT LineNumber FROM @bit WHERE output LIKE "Volume%";
SELECT
(SELECT SUBSTR(output, 8, 2) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumb
er)  AS DriveLetter,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
-3) AS DiskSpace,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
```

```
-4) AS BitlockerVersion,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
-5) AS ConversionStatus,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
-6) AS PercentageEncrypted,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
-7) AS EncryptionMethod,
(SELECT SUBSTR(output, 27) FROM @bit WHERE @baseLN.LineNumber = @bit.LineNumber
-8) AS ProtectionStatus
FROM @baseLN;
```

1.5.5.4.5 RECURSIVE CTEs

The following is an example of how the Agent language can use a SQLite CTE. The temp table can have any number of columns so long as they are matched by the same number of columns in each of the sub-selects.

Example - CTE

```
@factorials = SELECT * FROM (
  WITH RECURSIVE
  temp(n, nfact) AS (
    SELECT 0, 1
    UNION ALL
    SELECT n+1,
    (n+1)*nfact FROM temp WHERE n < 9)
  SELECT * FROM temp
);
```

1.5.6 Instruction flow control

Early versions of the Tachyon Agent Language (SCALE) were sequential, and deliberately did not include constructs for branching, but later versions introduced additional keywords for controlling instruction flow and termination:

- EVALUATE - allows an instruction to terminate early if the preceding statement did not yield any results; it is in effect "continue if data available".
- FOREACH - (new in v3.0.0) allows looping to iterate over each row within a @table and execute one or more statements per row
- IF/ELSE/ENDIF - (new in v3.3) provides conditional branching
- RETURN, NOCONTENT, ERROR, and NOTIMPLEMENTED - (new in v3.3) allows Early exit termination of the instruction.

1.5.6.1 Inconsistent branching results

The reason why SCALE did not originally support branching is because poor coding could return a different "shape" of data as its output. The following example works fine as it is because both the IF and ELSE statements contain the same number of columns, with the same column names, and

can be output as the results of the instruction, or saved in a @table for processing in later statements.

Example - ensuring IF and ELSE statements have the same shape

```
@n =
Network.GetConnections();
@c = SELECT COUNT(1), 1 AS X FROM @n GROUP BY X HAVING COUNT(1)>10;
@results = IF (@c)
       SELECT Count(1) AS Connections FROM @n;
       -- SELECT * FROM @n;
       -- SELECT RemoteAddress, RemotePort FROM @n;
ELSE
       SELECT "Not many connections here" AS Connections;
ENDIF;
```

However, if instead the IF statement used either of the commented lines, then it would return a different number of columns to the ELSE statement, depending on the number of network connections detected. One would produce multiple columns such as "IpAddress", "Port" and the other a single column "Connections". You could only use one of these shapes as the instruction schema, and Tachyon would then be unable to process the different shape data because it does not match the instruction schema.

> **Note**
>
> TIMS does not detect inconsistent branching results (*shape shifting*). Extra attention to code testing is required when using IF/ELSE/ENDIF and Early exit statements, which is good practice anyway.

1.5.6.2 Using "EVALUATE" to break on no output

If an error occurs while the Agent executes the instruction, execution of the whole instruction will terminate and return corresponding error details back to the Tachyon Server. See Exit Codes for more details on how error status is reported.

There is no support for exception handling (e.g. TRY/CATCH), nor is there support for transactional logic (e.g. COMMIT/ROLLBACK).

EVALUATE is the simplest form of flow control, and allows you to optimize your instruction and improve its readability, by avoiding unnecessary execution of subsequent statements.

For example:

Example - breaking on no results

```
@connections = Network.GetConnections();
@https = SELECT * FROM @connections WHERE RemotePort = 443;

EVALUATE;

Scripting.Run(Language: "PowerShell", LanguageVersion: "3.0",
              Script: "AuditCertificates.ps1");
```

In this example, we execute a PowerShell script if (and only if) we detect that there are any HTTPS connections detected (port 443); otherwise execution stops at the EVALUATE statement, and the instruction returns the result of "Success with no data". The AuditCertificates.ps1 PowerShell script may itself contain logic to handle the case of no active HTTPS connections, but by explicitly including an EVALUATE keyword the overhead of launching PowerShell is avoided.

Note that using EVALUATE inside a FOREACH loop will exit the entire instruction if the previous statement had returned no data; it will not just exit the loop. This behaviour may change in a future version of Tachyon.

1.5.6.3 Looping with FOREACH

The FOREACH construct – like an iterator in C++, or an enumerator in .NET - allows you to repeat a block of one or more statements for each row in a @table, and optionally save in a @table the output of the last statement in the block.

Syntax

```
FOREACH @newTable IN @existingTable
DO
   //block of statements using @newTable (SELECT statements, Methods and/or Fun
ctions)
   //optional 'early exit' termination
DONE;
```

FOREACH does not have any conditional flow control (eg. WHILE or BREAK) but this may change in a future version of Tachyon. However, you can cause the instruction to terminate early using EVALUATE, RETURN, NOCONTENT, ERROR, and NOTIMPLEMENTED..

Here's an example, which deletes any files bigger than 500k and also any ".tmp" files.

Example - looping with FOREACH

```
@files = FileSystem.GetFilesInFolder(Folder: "C:\\BigFiles");
@filesToDelete = SELECT FileName
                 FROM @files
                 WHERE FileSize > 500000
                 OR FileName LIKE "%.tmp";

EVALUATE;

FOREACH @f IN @filesToDelete
DO
   FileSystem.DeleteFileByName(FileName: @f.FileName);
DONE;
```

In this example, we build a @table containing the files we want to delete, and then, for each row in that @table, we create a new @table called "@f", and execute the code between the "DO" and "DONE" keywords. Since our new @table "@f" has exactly one row, we can pass its values as parameters to the DeleteFileByName method. In this way, we delete each of the selected files, one by one.

It is important to note what the "output" value of FOREACH is. The FOREACH statement will take each output of the inner block (in our example, the output of the call to DeleteFileByName) and combine the results together. The output of FOREACH can also be assigned to a @table if required.

For example, here is how we could get the SHA256 hash of each file in a folder:

Example - capturing the output of FOREACH

```
@files = FileSystem.GetFilesInFolder(Folder: "C:\\BigFiles");

@hashes = FOREACH @f IN @files
DO
   FileSystem.GetHash(FileName: @f.FileName, Algorithm: "SHA256");
DONE;

SELECT FileHash, FileName FROM @hashes;
```

With each successive call to the GetHash method, the output of the FOREACH statement is combined with the output of the method.

We could run *multiple* statements for each file, although this is less common. If multiple statements occur between the DO and DONE keywords, it is the output of the *last* of those statements which is merged into the overall result of the FOREACH statement.

1.5.6.4 Conditional branching using IF/ELSE/ENDIF

Allows conditional branching of instruction flow based on a condition.

The condition cannot be a SELECT statement, but can be one of the following:

- A @table, such as "*`@myTable`*"
- A method invocation, such as "*`Network.GetConnections()`*"
- A literal, such as "1" (although use of this is uncommon)

In each of the above cases, the condition (which represents a table of data) will evaluate to "true" – and will therefore pass the IF check – if that table has one or more rows. A table with zero rows (including tables which have an undefined structure) will evaluate to "false".

The IF condition is followed by a block of code to execute if the condition evaluates to true, and then a closing *`ENDIF`*. You may optionally include an *`ELSE`* statement with a corresponding block of code to execute if the condition evaluates to false.

1.5.6.4.1 Simple branching

The following simple example runs a different executable depending on whether the target Agent is running on a 32- or 64-bit operating system.

```
@device = Device.GetSummary();
@x64 = SELECT * FROM @device WHERE OsArchitecture = "x64";
@results = IF (@x64)
    NativeServices.RunCommand(CommandLine: "my-64-bit-process.exe");
ELSE
```

```
    NativeServices.RunCommand(CommandLine: "my-32-bit-process.exe");
ENDIF;
```

Note that the expression to test – in preceding case @x64 – needs to be enclosed in parentheses.

In this case, because both the IF and ELSE statements run the same command, the results will have the same "shape" and can be saved in a @table for later processing.

As described earlier, the *ELSE* is optional – so the following example runs an executable if and only if the target Agent is running on a 64-bit operating system (but will do nothing on 32-bit operating systems).

```
@device = Device.GetSummary();
@x64 = SELECT * FROM @device WHERE OsArchitecture = "x64";

IF (@x64)
    NativeServices.RunCommand(CommandLine: "my-64-bit-process.exe");
ENDIF;
```

The body of an *IF* block (or an *ELSE* block) can contain multiple statements, such as in the following example:

```
@device = Device.GetSummary();
@x64 = SELECT * FROM @device WHERE OsArchitecture = "x64";

IF (@x64)
    NativeServices.RunCommand(CommandLine: "my-first-64-bit-process.exe");
    NativeServices.RunCommand(CommandLine: "my-second-64-bit-process.exe");
ENDIF;
```

1.5.6.4.2 Nesting

IF blocks can also be nested; the following example shows running an executable if the target machine is running on Windows, and a different executable depending on the operating system is 32- or 64-bit:

```
@device = Device.GetSummary();

@isWindows = SELECT * FROM @device WHERE OperatingSystemType = "Windows";
@x64 = SELECT * FROM @device WHERE OsArchitecture = "x64";

IF (@isWindows)
   IF (@x64)
       NativeServices.RunCommand(CommandLine: "my-64-bit-process.exe");
   ELSE
       NativeServices.RunCommand(CommandLine: "my-32-bit-process.exe");
   ENDIF;
ELSE
   // Other logic to handle other operating systems
ENDIF;
```

IF blocks can also contain any other statements, such as FOREACH loops and SELECT statements.

Warning

SCALE does not presently support an "*ELSEIF*" keyword; to emulate this behaviour (e.g. if A then do X, else if B then do Y, else if C then do Z, etc.) you can use the nesting approach described above.

1.5.6.4.3 IF Conditions

The previous examples all use the result of a *SELECT* statement on the result of a method call as their expression in the IF condition. You can however put the method call, including any required parameters, directly in the *IF* condition. The following example will run a command if a specified file exists:

```
IF (FileSystem.GetFileDetails(FilePath: "c:\\MyFolder\\MyFile.dat"))
    NativeServices.RunCommand(CommandLine: "process-my-file.exe");
ENDIF;
```

The example above works because *FileSystem.GetFileDetails* will return no rows (and therefore the condition will evaluate to false) if the file does not exist, or will return a single row (in which case the condition is true) if the file does exist.

Warning

SCALE does not presently support using operators or *SELECT* statements directly as the IF condition; you must instead perform the *SELECT* before the IF, assigning the result to a @table, and then use the @table as the condition.

Lastly, the expression may be a literal value, such as a string (enclosed within double-quotes) or a number. Any literal value – including NULL – evaluates to a table with a single column (named "Value") and a single row (with the literal value), so will always evaluate to true. This can be useful for testing code paths during development of an instruction – in other words, you may wish to temporarily force a condition to be true.

```
IF ("hello")
    Agent.Echo(Message: "This will always be executed");
ENDIF;

IF (1)
    Agent.Echo(Message: "So will this");
ELSE
    Agent.Echo(Message: "And this will NEVER be executed");
ENDIF;
```

Tip

This syntax is generally useful only for testing purposes; you would be unlikely to use this form of *IF* in a live instruction.

1.5.6.4.4 Negating an IF

You can use the *NOT* keyword in conjunction with *IF* to negate the expression being tested; for example:

```
@procs = OperatingSystem.GetProcesses();
@oneDrive = SELECT * FROM @procs WHERE Executable LIKE "OneDrive.exe";

IF NOT (@oneDrive)
    // OneDrive is not running - take further action here
ENDIF;
```

In the example above, the IF block is executed if @oneDrive contains no rows (which it won't if there is no running process whose executable name is like "OneDrive.exe".

> **Note**
>
> The *NOT* operator *occurs* outside of the parentheses, and not inside. Technically speaking, the "*NOT*" does not form part of the expression, but is actually an extended form of the *IF* statement.

1.5.6.4.5 Capturing the result of an IF statement

Like *FOREACH* loops, *IF* statements themselves are an expression, in that they yield a value. The value returned by an *IF* statement is the last result of the block of statements executed. This can then be assigned to a @table, for example:

```
// Get lines from configuration file if it exists
@lines = FileSystem.GetFileByLine(FilePath: "C:\\MyFolder\\MyConfiguration.txt"
);

@setting = IF (@lines)
    // Configuration file exists - get the setting from the content
    // of line number 10
    SELECT Content AS MySetting FROM @lines WHERE LineNumber = 10;
ELSE
    // Configuration file does not exist (no lines were read)
    // so fall back to reading from registry
    @reg = NativeServices.RegistryGetValue(
        Hive:"HKLM", Subkey:"SOFTWARE\\MyVendor\\MyConfiguration",
        Name:"MySetting");

    // Append default domain suffix to whatever we have read
    SELECT Value || ".acme.local" AS MySetting FROM @reg;
ENDIF;

// Do something with the setting
SELECT "My setting is || " MySetting AS Value FROM @setting;
```

In this example, the @setting table is assigned the result of the *IF* statement, which will be either the result of SELECTing from @lines or from @reg, depending on the path of execution.

The same result could also be achieved by having both the *IF* and the *ELSE* branch of the code assign to the same @table:

```
@lines = FileSystem.GetFileByLine(FilePath: "C:\\MyFolder\\MyConfiguration.txt"
);
```

```
IF (@lines)
    @setting = SELECT Content AS MySetting
               FROM @lines
               WHERE LineNumber = 10;
ELSE
    @reg = NativeServices.RegistryGetValue(
        Hive:"HKLM", Subkey:"SOFTWARE\\MyVendor\\MyConfiguration",
        Name:"MySetting");

    @setting = SELECT Value || ".acme.local" AS MySetting FROM @reg;
ENDIF;

SELECT "My setting is || " MySetting AS Value FROM @setting;
```

1.5.6.4.6 Different data from different branches

Something to be careful of when using *IF* statements is having the IF branch of the code return data which is a different "shape" (i.e. different columns) to the *ELSE* branch of the code.

Consider the following:

```
@result = IF (@myTable)
    SELECT A, B, C FROM @myTable;
ELSE
    SELECT X, Y, Z FROM @myOtherTable;
ENDIF;

// Problem! - @result will have different columns depending on
// which branch of the IF statement was executed
```

In this example, depending on the path taken, the @result table will either have columns called A, B and C, or will have X, Y and Z. This means that subsequent code which uses @result may fail.

You should take steps to avoid this situation. If you intend to use the result of an *IF* statement, ensure that the column names (and datatypes) are consistent between the *IF* and *ELSE* branches.

1.5.6.4.7 Do you really need to use an IF?

IF statements are useful for executing different code paths depending on some condition – for example, one branch of your code might have to read some registry values and manipulate them, while the other branch may need to execute a command-line or run a WMI query.

However, often you may simply want to deal with different *data* in each code branch, but what you do with that data may be identical.

In that case, it can sometimes be easier to express this logic in SQL rather than using an *IF* statement – especially if you need to deal with multiple conditions.

Consider the case where you want to execute a different command depending on the Agent's operating system; you could use a series of *IF*/*ELSE* statements to perform this, but an easier way to achieve the result could be as follows:

```
@device = Device.GetSummary();
@cmd = SELECT CASE OperatingSystemType
              WHEN  "Windows" THEN "my-windows-command-line.exe -a -b -c"
              WHEN  "MacOS"   THEN "/my/mac/command-line -d -e -f"
                            ELSE  "unsupported"
              END AS CommandToExecute
       FROM @device;

NativeServices.RunCommand(CommandLine: @cmd.CommandToExecute);
```

The SQL above builds a different command-line (using the *SELECT CASE* construct) depending on the operating system type, and then runs that command.

1.5.7 Advanced concepts

1.5.7.1 Passing @tables as method parameters

As your instructions become complex, you may wish to pass parameters to methods using @tables, rather than fixing the values. A common scenario of this is FOREACH – described later.

Consider the following example:

Example - passing @tables as parameters

```
/* Get all the files in the C:\BigFiles folder */
@files = FileSystem.GetFilesInFolder(Folder: "C:\\BigFiles");

/* Bail out if there are no files */
EVALUATE;

/* Pick the biggest one */
@biggest = SELECT FileName FROM @files ORDER BY FileSize DESC LIMIT 1;

/* And then delete it */
FileSystem.DeleteFileByName(FileName: @biggest.FileName);
```

This example deletes the biggest file in a specific folder. We use LIMIT 1 to get the biggest file ordered by FileSize, and then pass that value to the DeleteFileByName method.

However, this syntax has a specific constraint. In our example, the use of LIMIT 1 guarantees that our "biggest" @table contains exactly one row. This is important, because parameters to methods (such as "DeleteFileByName" in this case) cannot take a set of values – they take only one.

If we had removed the LIMIT 1 clause from our query (and we had more than one file in our C:\BigFiles folder), this instruction would fail. This is because we are attempting to pass multiple "rows" of values for a single parameter.

To summarize – a field (column) from a @table can be passed as a value for parameter if:

- The @table contains *exactly one row*, and
- The field from the @table is of the correct datatype for the parameter

Additionally, you can omit the field name from the @table if that @table contains *exactly one field*:

Example - passing non-qualified @tables as parameters

```
@files = FileSystem.GetFilesInFolder(Folder: "C:\\BigFiles");

EVALUATE;

@biggest = SELECT FileName FROM @files ORDER BY FileSize DESC LIMIT 1;

/* Valid - @biggest contains only one field (FileName) so no need to specify it
*/
FileSystem.DeleteFileByName(FileName: @biggest);

/* ... however... */

@smallest = SELECT FileName, FileSize FROM @files ORDER BY FileSize ASC LIMIT 1
;

/* NOT valid - @smallest contains two fields (FileName + FileSize) */
// FileSystem.DeleteFileByName(FileName: @smallest);

/* But this is fine */
FileSystem.DeleteFileByName(FileName: @smallest.FileName);
```

If you cannot guarantee that your @table contains exactly one row, you will need to use the FOREACH construct to iterate over the @table.

1.5.7.2 Agent SQL functions

SQLite includes a number of built-in functions which you can use in SELECT statements:

- Core functions, such as IFNULL, INSTR, SUBSTR, REPLACE, TRIM, UPPER, LOWER, RANDOM, PRINTF etc. are documented online at: https://sqlite.org/lang_corefunc.html.
- Aggregate functions, such as COUNT, MIN, MAX, GROUP_CONCAT etc. documented online at: https://sqlite.org/lang_aggfunc.html.
- Date and time functions, documented online at: https://sqlite.org/lang_datefunc.html. See datetime handling.

The Tachyon Agent takes advantage of SQLite extensibility by providing a further "application defined functions" that are built-in to the Agent and can also be used in SELECT statements.

- Page:BASENAMEFROMPATH— Takes a path as a string and returns the basename.
- Page:BASENAMEFROMPATH— Takes a path as a string and returns the basename.
- Page:COMBINEPATHS— Takes two paths as strings, or a path and a basename as strings. It combines them and returns the result.

- Page:COMBINEPATHS— Takes two paths as strings, or a path and a basename as strings. It combines them and returns the result.
- Page:COMPAREVERSIONS— Compare two version strings.
- Page:COMPAREVERSIONS— Compare two version strings.
- Page:DATETRUNC— Truncates a date-time value, expressed as a Unix epoch time integer, to a given resolution and returns a corresponding Unix epoch time integer.
- Page:DATETRUNC— Truncates a date-time value, expressed as a Unix epoch time integer, to a given resolution and returns a corresponding Unix epoch time integer.
- Page:DIRECTORYFROMPATH— Takes a path as a string and returns the parent directory of the basename.
- Page:DIRECTORYFROMPATH— Takes a path as a string and returns the parent directory of the basename.
- Page:EPOCHTOJSON— Takes a date-time value, expressed as a Unix epoch time in UTC, and returns a JSON-compatible (ISO-8601) string representation of that date-time, also in UTC.
- Page:EPOCHTOJSON— Takes a date-time value, expressed as a Unix epoch time in UTC, and returns a JSON-compatible (ISO-8601) string representation of that date-time, also in UTC.
- Page:EXPAND— Expands environment variables in a string
- Page:EXPAND— Expands environment variables in a string
- Page:HASHSTRING— Produce a hash of a string value based on a named hashing algorithm
- Page:HASHSTRING— Produce a hash of a string value based on a named hashing algorithm
- Page:IPINRANGE— Determines if an IP Address is within a specified range or ranges.
- Page:IPINRANGE— Determines if an IP Address is within a specified range or ranges.
- Page:IPV4STRTOINT— Converts an IPv4 address into an integer.
- Page:IPV4STRTOINT— Converts an IPv4 address into an integer.
- Page:NEWID— Generates a new universally unique identifier (also known as a GUID)
- Page:NEWID— Generates a new universally unique identifier (also known as a GUID)
- Page:REGEXP— Determines if a string matches a regular expression.

- Page:REGEXP— Determines if a string matches a regular expression.
- Page:REGEXPCAPTURE— Uses a regular expression to capture matching parts from a string.
- Page:REGEXPCAPTURE— Uses a regular expression to capture matching parts from a string.
- Page:REGEXPREPLACE— Uses a regular expression to replace a string.
- Page:REGEXPREPLACE— Uses a regular expression to replace a string.

1.5.7.3 Agent built-in functions

The functions described above are SQL-based functions, so can be used in SELECT statements. The Agent language itself also offers some built-in functions used outside of SELECT statements – think of them like methods available in a "global" module.

- Page:HttpGetFile— A built-in function for downloading content (resources) over HTTP(S) via the Tachyon Background channel, or from an external web-server.
- Page:HttpGetFile— A built-in function for downloading content (resources) over HTTP(S) via the Tachyon Background channel, or from an external web-server.
- Page:IsUndefined— Given a table as a parameter, this built-in function will output a table that indicates whether the input table is real and full of values or whether it is the result of an operation that produced nothing, which is different to producing an empty table.
- Page:IsUndefined— Given a table as a parameter, this built-in function will output a table that indicates whether the input table is real and full of values or whether it is the result of an operation that produced nothing, which is different to producing an empty table.
- Page:Raise— A built-in function to raise one or more events.
- Page:Raise— A built-in function to raise one or more events.

1.5.7.4 Global Settings variables

Tachyon provides an ability to store variables centrally, in the master database, and use those variables in instructions, in the Readable Payload or within the instruction (Payload).

They are similar to instruction parameters but do not prompt for a value in Tachyon Explorer. They conform to following pattern: %global:variable% where variable would be the name of the variable you want to use. All variables are case-insensitive.

So if in your Global Setting table you have an entry with the name 'ExecutionPolicy' you can use this in the payload and/or readable payload: %global:ExecutionPolicy% and the Value present for that entry will be inserted into the payload(s) replacing the %global:ExecutionPolicy%.

Please note the setting you request must exist in the database in order for the instruction to be issued, as an error will be returned if you attempt to issue an instruction based on a definition that uses a Global Setting that cannot be found.

The Tachyon Settings application does not (yet) provide a means of managing Global Settings, therefore you must edit the database directly.

In TIMS, when you are testing an instruction that uses Global Settings variables, TIMS will prompt you to enter a value. This is because TIMS does not have access to a Tachyon System.

> **Warning**
>
> Do not use Global settings variables in Guaranteed State fragments. They are not supported. Even if they could be used, the Value would be inserted into the policy payload at the point it is deployed, and any later changes to the Value would not be used.
>
> Instead, Guaranteed State fragments should use a normal parameter, and the value for the parameter specified when configuring rules in Guaranteed State administration UI.

1.5.7.5 Environment variables

They are similar to instruction parameters but conform to following pattern: %environment:variable% where variable would be the name of the variable you want to use. All variables are case-insensitive.

If you use one of the patterns below in the Readable Payload or within the instruction (Payload), it will be replaced with another value. The 'other' value depends on which pattern you specified.

These following variables are available and hard coded in the system. They are not extensible in any way and each has a fixed behaviour.

Variable pattern	Replaced with	Introduced
%environment:now%	UTC ISO-8601 compatible date time stamp of when the instruction was issued. **Note** This is the timestamp of when the instruction was originally run and sent by the Tachyon Server. It remains the same if the same instruction is re-run.	**Tip** v3.1

1.5.7.6 Querying historic data

The Tachyon Agent has the ability to capture security-related events from the underlying operating system and store/summarize the data in a local database.

The feature is described here: Tachyon Activity Record.

The tables used to store this data are available to be queried from within the Agent Language using a slightly modified syntax. The related tables are available using the dollar ($) prefix.

For example:

Example - querying historic captured data

```
/* Sum the number of connections made per process today */
SELECT    SUM(ConnectionCount) AS Connections
,         ProcessName
FROM      $TCP_Daily
WHERE     TS = DATETRUNC(STRFTIME("%s", "now"), "day")
GROUP BY  ProcessName;
```

1.5.8 Instruction termination

1.5.8.1 Exit Codes

The Agent returns one of these codes to the Tachyon Server when it executes an instruction.

If a statement is successful it will return an Exit Code of 0 and a response which can be seen in the Content tab of the Explorer Responses page.

If a statement has no content, or produces an error, then no response is seen in the Content tab, however the Status tab of the Explorer Responses page shows a count for each Exit Code, and the ErrorData message where applicable.

If a statement exits early or produces an error then execution of the instruction terminates immediately with an Exit Code. Any changes made by the Agent are not rolled back.

Status	Code	ErrorData message	Meaning
Success	0	n/a	Success with content. This exit code is returned if the instruction is successful and has content. **V3.3** onwards lets you return 0 when a RETURN

Status	Code	ErrorData message	Meaning
			statement is used.
Success - no content	1	n/a	Success with no content. This exit code is returned if the instruction is successful but has no content. This can occur if an EVALUATE statement is used. **V3.3** onwards lets you return 1 when a RETURN , EVALUATE ,or NOCONTENT statement is used.
Error	2	Missing or invalid parameter 'xxx'.	The parameters do not match in quantity or type that the method expects. For

<table>
<tr><th>Status</th><th>Code</th><th>ErrorData message</th><th>Meaning</th></tr>
<tr><td></td><td></td><td></td><td>example, a mandatory parameter name is missing, or its case-sensitive name is not correct.</td></tr>
<tr><td>Execution of SELECT failed.</td><td>A SELECT statement is not constructed correctly.

The Tachyon Agent language processor does not fully parse or check the syntax of the SELECT expression, but instead just passes it on (with variable name substitutions and string literal translations) to the SQLite API. Hence, if there is a syntax error, any line or column numbers in an error message are not precise.</td><td></td><td></td></tr>
<tr><td>Syntax error in instruction.</td><td>A parameter value is not correct.

An escape character may be required.

A statement may not be constructed correctly. For example, a ; terminator or other part of a statement is missing; or has superfluous characters.</td><td></td><td></td></tr>
<tr><td><method specific errors></td><td>Some examples:

<table><tr><th>Method</th><th>ErrorData</th><th>Notes</th></tr><tr><td>NativeServices.RunCommand</td><td>Command line returned exit code nnn.</td><td></td></tr><tr><td>Scripting.Run</td><td>'Language' returned</td><td>'Language' is either Powershell or bash.</td></tr></table></td><td></td><td></td></tr>
</table>

Status	Code	ErrorData message	Meaning
	exit code nnn:		
Missing 'xxx' provider.	A supported method or function is used but is unable to find the required provider. The provider file does not exist on the device.		
<custom error message>	**V3.3** onwards lets you return custom messages as defined in an ERROR statement.		
Not implemented	3	No built-in function named 'xxx'.	The named function does not exist. This error is usually caused by a typo outside of a SELECT statement.
Unsupported module 'xxx'.	The module is not supported on this platform. This error can occur if: • the module is not enabled, for example Nomad methods are not available if Module.Nomad.Enabled=false • the module is not supported on this platform • the module library file does not exist on the device • the case-sensitive module name is not specified correctly		
Unsupported method 'xxx'.	The method is not supported on this platform. This error can occur if: • the method is not supported on this platform		

Status	Code	ErrorData message	Meaning
	• the case-sensitive module name is not specified correctly		
Script language 'xxx' is not supported.	The language specified in Scripting.Run(Language: "xxx" ....) is not supported on this platform. This error can occur if: • the language is not supported on this platform (PowerShell is supported on Windows, bash is supported on Linux and MacOS) • the language name is not specified correctly. Language name is not case-sensitive.		
<custom error message>	**V3.3** onwards lets you return custom messages as defined in a NOTIMPLEMENTED statement.		
Response too large	4		

1.5.8.2 Early exit

The EVALUATE keyword allows an instruction to terminate early if the preceding statement did not yield any results.

From v3.3, SCALE supports the following additional termination keywords which can be used to exit an instruction, and return a corresponding Exit Code. These are most useful when used in conjunction with *IF/ELSE/ENDIF* construct but can be used anywhere within an instruction.

- RETURN (0/1)
- NOCONTENT (1)
- ERROR (2)
- NOTIMPLEMENTED (3)

1.5.8.2.1 RETURN

RETURN allows you to exit out of an instruction early with a given set of results.

Syntax

```
@results = "some statement";
RETURN @results;
```

Ordinarily, the results of an instruction are based on the last statement to execute; RETURN allows you to do this at any point in an instruction. In contrast to EVALUATE (which means “break out of this instruction if the preceding statement returned no results), *RETURN* breaks unconditionally. With *RETURN*, you supply the @table whose value should become the results of the instruction.

The instruction exits successfully, with an exit code of 0 if the results have content, and 1 if they have no content.

Here’s an example:

```
@reg = NativeServices.RegistryGetValue(
            Hive:"HKLM",
            Subkey:"SOFTWARE\\MyVendor\\MyConfiguration",
            Name:"SecuritySetting");

// If the setting exists, return a message indicating we're secure
IF (@reg)
    @msg = SELECT "System already secure" AS Status;
    RETURN @msg;
ENDIF;

// Otherwise, take some action...
// ... and report that back. No need to use RETURN, because
// this is the last statement in the instruction.
SELECT "System was insecure, but is now secure" AS Status;
```

1.5.8.2.2 NOCONTENT

Syntax

```
//some statements
NOCONTENT;
```

NOCONTENT is an unconditional version of the *EVALUATE* keyword. Each causes an instruction to exit successfully, with an exit code of 1 meaning “Success, no content”, and do not return a message.

Here’s an example:

```
@users = Users.GetLoggedOnUsers();

// Return "success no content" if they are any logged in users
IF (@users)
    NOCONTENT;
ENDIF;

// ... otherwise do something which can only be done if no-one is logged on
```

1.5.8.2.3 ERROR

Syntax

```
//some statements
ERROR "custom error message";
```

ERROR causes an instruction to terminate with exit code of 2 meaning “Error” and the specified custom error message. The error message must be supplied as a string literal, and cannot be a reference to a @table or any other kind of expression.

Here’s an example:

```
@svcs = OperatingSystem.GetServiceInfo();
@nomad = SELECT * FROM @svcs WHERE Name = "NomadBranch";

IF NOT (@nomad)
    ERROR "1E Nomad is not available";
ENDIF;

// Otherwise do some work with 1E Nomad...
```

1.5.8.2.4 NOTIMPLEMENTED

Syntax

```
//some statements
NOTIMPLEMENTED "custom error message";
```

NOTIMPLEMENTED causes an instruction to terminate with exit code of 3 meaning “Not implemented” and the specified custom error message. The error message must be supplied as a string literal, and cannot be a reference to a @table or any other kind of expression.

Here’s an example:

```
@dev = Device.GetSummary();
@win = SELECT * FROM @dev WHERE OperatingSystemType = "Windows";

IF NOT (@win)
    NOTIMPLEMENTED "This functionality is only available on Windows";
ENDIF;

// ... otherwise do something Windows-specific
```

(END OF PAGE)

1.5.9 Datetime handling

Guidance and examples for using date and time values in instructions.

1.5.9.1 Datetime Functions

In addition to those functions provided by SQLite, the Agent language has the following date functions for use in SELECT statements:

- DATETRUNC — Truncates a date-time value, expressed as a Unix epoch time integer, to a given resolution and returns a corresponding Unix epoch time integer.

- EPOCHTOJSON — Takes a date-time value, expressed as a Unix epoch time in UTC, and returns a JSON-compatible (ISO-8601) string representation of that date-time, also in UTC.

1.5.9.2 Formats

Irrespective of the local timezone or daylight savings adjustment, the Agent always presents datetime as UTC unless a function is used to specifically convert to local time.

If you want to know the device's timezone, use the Agent.GetSummary method, which reports current TimeZone as an offset from UTC in minutes (positive or negative). If using TIMS, then you will need to restart TIMS to read any changes in timezone.

Agent methods return datetime columns as either

- UnixEpoch, defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, which may be either SQL datatype integer or string
- ISO-8601 string YYYY-MM-DDTHH:MM:SSZ in which the timezone offset is always Z because Agent time is always presented in UTC.

1.5.9.2.1 Converting from Unix Epoch timestamp into readable formats

As previously stated, SQLite has many date and time functions, documented online at: https://sqlite.org/lang_datefunc.html. The Agent language also provides the EPOCHTOJSON function to convert a UnixEpoch integer to a ISO-8601 string. The example below shows how the SQLite functions STRFTIME and DATETIME could be used instead. It also shows these functions do not require the TS string to be CAST as an integer. Finally, the example also shows how DATETIME uses TimeZone.

Example used to display Unix Epoch timestamp in different formats

```
// Get TimeZone and current time
@summary = Agent.GetSummary();
@now = SELECT STRFTIME("%s","now") AS TS;
// Next statement shows that TS string can be CAST as an integer
@now = SELECT TS, CAST(TS AS INTEGER) AS TS_int, TimeZone FROM @now, @summary;
// Now show datatypes and conversions
SELECT
  TS,
  EPOCHTOJSON(TS) AS TS_EpochToJson,
  DATETIME(TS, "unixepoch") as TS_datetime,
  DATETIME(TS, "unixepoch", "localtime") as TS_localtime,
  STRFTIME("%Y-%m-%dT%H:%M:%SZ",DATETIME(TS, "unixepoch")) as TS_strftime,
  TYPEOF(TS),
  TYPEOF(TS_int),
  TimeZone
FROM @now;
```

The above example instruction will return something similar to the following in TIMS.

TS	TS_EpochToJson	TS_datetime	TS_localtime	TS_strftime	TYPEOF(TS)	TYPEOF(TS_int)	TimeZone
1515972745	2018-01-14T23:32:25Z	2018-01-14 23:32:25	2018-01-14 18:32:25	2018-01-14T23:32:25Z	text	integer	-300

1.5.9.2.2 Converting from readable format into Unix Epoch timestamp

Example used to convert ISO-8601 formatted string into Unix epoch time

```
// Converts datetime string "2017-08-18T19:18:03Z" or "2017-08-18 19:18:03" int
o unixepoch integer 1503083883.
// STRFTIME returns the integer as a TEXT datatype; if we want NUMERIC datatype
then we need to CAST
@unixepoch = SELECT CAST(STRFTIME("%s","2017-08-18T19:18:03Z") AS INTEGER) AS T
S;
```

1.5.9.2.3 Making data lists

Example to create list of last 30 dates

```
@now = SELECT STRFTIME("%s","now") AS TS;
@seq = Utilities.GenerateSequence(Limit: 30);
FOREACH @i IN @seq
DO
    @then = SELECT @now.TS-((@i.Sequence-1)*86400) AS TS FROM @now, @i;
    SELECT STRFTIME("%Y-%m-%d",DATETIME(TS, "unixepoch")) AS Date FROM @then;
DONE;
```

Example to create list between two dates

```
@dates = SELECT "2018-01-10" AS D1, "2018-01-14" AS D2;
@days = SELECT CAST(JULIANDAY(D2) - JULIANDAY(D1) + 1 AS INTEGER) AS Diff FROM
@dates;
@seq = Utilities.GenerateSequence(Limit: @days.Diff);
FOREACH @i IN @seq
DO
    SELECT STRFTIME("%Y-%m-%d", (JULIANDAY(D1) + @i.Sequence-1)) AS Date FROM @
dates, @i;
DONE;
```

1.5.9.3 When is now?

The examples above show how STRFTIME can tell you when an instruction was run, as UTC and adjusted to local time using TimeZone.

Sometimes it is useful for an instruction to know when it was originally sent, which may have been many moments before it was run. SCALE provides this with the following Environment variable:

%environment:now%	UTC ISO-8601 compatible date time stamp of when the instruction was issued.

> **Note**
>
> This is the timestamp of when the instruction was originally run and sent by the Tachyon Server. It remains the same if the same instruction is re-run.

Example to show difference of when an instruction is run and when it was sent

```
// these times are the same in TIMS
SELECT EPOCHTOJSON(STRFTIME("%s","now")) AS TS_run, "%environment:now%" AS TS_s
ent;
```

1.5.10 Client Activity Record

Reference information about the feature, sometimes referred to as either the inventory, or forensics feature, and previously known as Agent Historic Data Capture.

1.5.10.1 What is Client Activity Record?

The Tachyon client, while running, continuously captures details of certain activities and events as they happen, similar to Windows Task Manager or Perfmon. During startup, the Tachyon client is able to detect some events that occurred when it was not running. Data is regularly written into a local, compressed and encrypted **persistent storage tables**, that are accessible to SCALE as SQL tables. The Tachyon client periodically aggregates data in order to minimize the amount of storage required, so that each capture source has a live, hourly, daily and monthly table. The whole process is designed to minimize impact on device performance, storage and security.

Persistent storage tables cannot be deleted or modified because they are managed by the Tachyon client and used by the Tachyon Activity Record feature. Whereas User Defined Persistent Storage Tables can be created, deleted and modified using SCALE because they are managed by Tachyon instructions run by users.

The Client Activity Record schema is provided below.

> **Note**
>
> To use any TAR features you must have Inventory enabled by setting **Module.Inventory.enabled=true** in the client configuration file. This setting, and configuration options for each capture source are described in 1E Client 8.1 - Tachyon client settings: Capture source settings.
>
> With some exceptions, the defaults are the same for each capture source, but aggregation and retention settings can be set individually for each table.

The Tachyon client has two mechanisms of knowing when an event occurs that is of interest:

- **Event-based** relies on a source external to the Tachyon client (normally the operating system) providing a notification to indicate that something has happened

- **Polling-based** is where the Tachyon client will periodically check a source of data and work out what has changed by looking at differences in the data returned. Polling intervals means some brief events that occur between polls can be missed.

On Windows, the Tachyon client is able to use Event Tracing for Windows (ETW). However, if desired, the individual capture sources can be configured to use polling instead of ETW.

Other data collection methods that run periodically (polling-based) for a short period:

- Windows performance counters for disk, memory, network and processor performance
- a proprietary **metric** collection process that tests the Operating System and its software.

Note

Starting with Tachyon 5.1, the 1E Client UI component (which runs in each user's session) provides data to the Tachyon client about the user's interaction with the device and foreground applications. This data is available by querying the Device interaction and Software interaction tables.

For this data to be captured, you must have Interaction enabled (set **Module.Interaction.Enabled=true**) as well as user interaction sampling (set **Module.Interaction.SampleUserInteraction=true**). Both are enabled by default.

For more details on configuring the Interaction module, see 1E Client 8.1 - Tachyon client settings: Interaction module settings.

1.5.10.1.1 Capture sources

The table below lists currently supported data capture sources, on which OS they are supported, and which capture method is used by default. See Constraints of Legacy OS regarding Windows XP, Vista and Windows Server 2003.

TAR data source	Description	Required by	Windows	macOS	Linux
ARP cache entries $ARP_xxx	The Tachyon client captures translations between IP addresses and MAC (physical) addresses, known as ARP (Address Resolution Protocol).		• Introduced in 3.2 • Polling on all versions of Windows	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	ARP cache polling is every 30 seconds.				
Boot performance $BootPerformance_Live	The Tachyon client captures boot performance related metrics from events logged by Windows OS.	Experience	• Introduced in 8.0 • Windows Event Log	• Some features introduced in 8.2	*Not yet available*
Device interaction $DeviceInteraction_xxx	The Tachyon client captures user interaction (keyboard and mouse activity) with the device for each (local and remote) user session. Data is captured by the 1E Client UI, whose behavior is controlled by the Interaction module.	Experience	• Introduced in 5.1 • Data from the 1E Client UI	*Not yet available*	*Not yet available*
Device performance $DevicePerformance_xxx	The Tachyon client captures metrics for	Experience	• Introduced in 5.0	• Some features intr	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	device performance by interrogating Windows Performance Counters. These metrics cover disk, memory, network, and processor performance. Device performance polling is every 10 seconds.		• Windows Performance Counters	oduced in 8.2	
Device resource demand $DeviceResourceDemand_xxx	The Tachyon client captures high-level device resource allocation and utilization - specifically for CPU, disk, network and memory. This data is used by the VDI monitoring feature of Tachyon Experience to show resource usage	Experience	• Introduced in 5.1 • Windows Performance Counters	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	patterns, and to help identify under- and over-provisioned virtual servers. Device resource demand polling is every 10 seconds.				
DNS resolutions $DNS_xxx	The Tachyon client captures whenever a DNS address is resolved. When using the polling method, the polling interval is every 30 seconds.		• Introduced in 2.1 • Polling on Windows 8 and below • ETW on Windows 8.1 and above	*Withdrawn in 5.2*	*Not yet available*
Operating System performance	The Tachyon client captures metrics for	Experience	• Introduced in 5.0	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
$OperatingSystemPerformance_xxx	performance and sensitive processes by running a metrics executable every four hours by default, that captures 15 performance metrics. Operating system performance polling is every 4 hours (14,400 seconds).		• Proprietary metric collection		
Performance event $PerformanceEvent_xxx	The Tachyon client captures a summary of operating system events which relate to device performance and stability, such as application/operating system crashes, patch installation and	Experience	• Introduced in 5.1 • Windows Event Log	• Some features introduced in 8.2	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	uninstallation, etc. Events are captured as they are generated by the operating system.				
Process executions $Process_xxx	The Tachyon client captures whenever a process starts on the device. When using the polling method, the polling interval is every 30 seconds.		• Introduced in 2.1 • Polling on Windows XP • ETW on Windows Vista and above	• Introduced in 2.1 • Polling	• Introduced in 2.1 • Polling
Process stabilizations $ProcessStabilization_xxx	The Tachyon client captures the time taken for a process to be considered stable. This is captured when a process starts on a device,		• Introduced in 3.2 • ETW on Windows Vista and	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	provided that process is in a list of processes selected for monitoring in the 1E Client configuration file.		above		
Process usage $ProcessUsage_Daily	The Tachyon client captures details about running processes from start to end. When using the polling method, the polling interval is every 30 seconds.	Tachyon Powered Inventory	• Introduced in 3.2 • Polling on Windows XP • ETW on Windows Vista and above	*Not yet available*	*Not yet available*
Sensitive processes $SensitiveProcess_xxx	The Tachyon client captures metrics for performance and sensitive processes by running a metrics executable every four hours by	Experience	• Introduced in 5.0 • Proprietary metric collection	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	default, that captures details for each sensitive process. Sensitive processes polling is every 4 hours (14,400 seconds).				
Software installations $Software_xxx	The Tachyon client captures which software is present on a device, and when it is installed and uninstalled. Software polling is every 120 seconds.		• Introduced in 2.1 • Polling on all versions of Windows	• Introduced in 2.1 • Polling	• Introduced in 2.1 • Polling
Software interaction $SoftwareInteraction_xxx	The Tachyon client captures user interaction (keyboard and mouse activity) with each application that enters the foreground.	Experience	• Introduced in 5.1 • Data from the 1E Client UI	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	Data is captured by the 1E Client UI, which is controlled by the Interaction module.				
Software performance $SoftwarePerformance_xxx	The Tachyon client captures metrics for software performance in terms of disk I/O, memory and processor usage. Software performance polling is every 10 seconds.	Experience	• Introduced in 5.0 • Windows Performance Counters	• Introduced in v8.2	*Not yet available*
SoftwarePerformance.DiskUsage $SoftwarePerformance_xxx	The Tachyon Client captures details about disk i/o operations for all running processes. We register 'Microsoft-Windows-Kernel-Disk' Event tracing	Experience	• Introduced in 8.0 • Event tracing	*Not yet available*	*Not yet available*

TAR data source	Description	Required by	Windows	macOS	Linux
	provider to gather data				
SoftwarePerformance.ProcessNetworkUsage $SoftwarePerformance_xxx	The Tachyon Client captures details about network activity for all running processes. We use a real time 'NT Kernel Logger' event tracing session to gather network traffic data	Experience	• Introduced in 8.0 • Event tracing	*Not yet available*	*Not yet available*
TCP outbound connections $TCP_xxx	The Tachyon client captures whenever an outbound TCP connection is made. When using the polling method, the polling interval is every 30 seconds.		• Introduced in 2.1 • Polling on Windows XP • ETW on Windows Vista and above	• Introduced in 2.1 • Polling	• Introduced in 2.1 • Polling

TAR data source	Description	Required by	Windows	macOS	Linux
User usage $UserUsage_Daily	The Tachyon client captures details about user sessions from login to logout. System accounts, and accounts used to run services, are excluded. The polling interval is every 30 seconds.	Tachyon Powered Inventory	• Introduced in 3.2 • Polling on all versions of Windows	*Not yet available*	*Not yet available*

1.5.10.2 How do I retrieve the data from Tachyon client devices?

Live and aggregated Tachyon Activity Record data is stored in the following persistent storage tables. You can simply query these using SELECT statements.

TAR data source	Live tables	Hourly tables	Daily tables	Monthly tables
ARP cache entries	$ARP_Live	$ARP_Hourly	$ARP_Daily	$ARP_Monthly
Boot performance	**$BootPerformance_Live**	**n/a**	**n/a**	**n/a**
Device interaction	$DeviceInteraction_Live	$DeviceInteraction_Hourly	$DeviceInteraction_Daily	$DeviceInteraction_Monthly

TAR data source	Live tables	Hourly tables	Daily tables	Monthly tables
Device performance	$DevicePerformance_Live	$DevicePerformance_Hourly	$DevicePerformance_Daily	$DevicePerformance_Monthly
Device resource demand	$DeviceResourceDemand_Live	$DeviceResourceDemand_Hourly	$DeviceResourceDemand_Daily	$DeviceResourceDemand_Monthly
DNS resolutions	$DNS_Live	$DNS_Hourly	$DNS_Daily	$DNS_Monthly
Operating System performance	$OperatingSystemPerformance_Live	$OperatingSystemPerformance_Hourly	$OperatingSystemPerformance_Daily	$OperatingSystemPerformance_Monthly
Performance event	$PerformanceEvent_Live	$PerformanceEvent_Hourly	$PerformanceEvent_Daily	$PerformanceEvent_Monthly
Process executions	$Process_Live	$Process_Hourly	$Process_Daily	$Process_Monthly
Process stabilizations	$ProcessStabilization_Live	$ProcessStabilization_Hourly	$ProcessStabilization_Daily	$ProcessStabilization_Monthly
Process usage	**n/a**	**n/a**	**$ProcessUsage_Daily**	**n/a**
Sensitive	$SensitiveProcess_Live	$SensitiveProcess_Hourly	$SensitiveProcess_Daily	$SensitiveProcess_Monthly

TAR data source	Live tables	Hourly tables	Daily tables	Monthly tables
processes				
Software installations	$Software_Live	$Software_Hourly	$Software_Daily	$Software_Monthly
Software interaction	$SoftwareInteraction_Live	$SoftwareInteraction_Hourly	$SoftwareInteraction_Daily	$SoftwareInteraction_Monthly
Software performance	$SoftwarePerformance_Live	$SoftwarePerformance_Hourly	$SoftwarePerformance_Daily	$SoftwarePerformance_Monthly
TCP outbound connections	$TCP_Live	$TCP_Hourly	$TCP_Daily	$TCP_Monthly
User usage	**n/a**	**n/a**	**$UserUsage_Daily**	**n/a**

Example - querying historic captured data

```
/* Sum the number of connections made per process today */
SELECT  SUM(ConnectionCount) AS Connections
,     ProcessName
FROM   $TCP_Daily
WHERE   TS = DATETRUNC(STRFTIME("%s", "now"), "day")
GROUP BY ProcessName;
```

Note the below example uses LIKE because the inventory tables are not created with COLLATE NOCASE, and need to be queried in a case-sensitive fashion. If ProcessName = "chrome.exe" is used then it will not match "Chrome.exe" or "chrome.EXE".

Example - handling case-sensitivity

```
SELECT * FROM $Process_Live WHERE ProcessName LIKE "chrome.exe"
```

1.5.10.3 How is the data managed?

The Tachyon client automatically aggregates and grooms data in each inventory table, according to aggregation intervals and data retention settings which are configurable in the 1E Client configuration file.

- Default aggregation cycle interval is every 60 seconds, therefore it may take up to a minute before an event appears in an aggregated table
- Default retention for live tables is 5000 entries provided at least 3 aggregation cycles have occurred (older entries are deleted to make room for new entries)
- Default retention for hourly tables is a rolling 24 hours.
- Default retention for daily tables is a rolling 31 days.
- Default retention for monthly tables is a rolling 12 months.

Each aggregated table is built from the live table, and does not have a dependency on other aggregated tables. For example, Monthly is fed by Live, not fed by Daily. This allows retention settings to be configured independently for each table.

Data is stored in a local, compressed and encrypted persistent store, which persists during a Tachyon client upgrade, uninstall and re-installation, unless specifically deleted.

If the Tachyon client is unable to write to storage (out of disk space or other file-system problems), it will fail but continue monitoring in the hope this situation will improve later.

1.5.10.4 Client Activity Record schema

The following table shows the fields which exist only in the Live and Aggregated (Hourly, Daily, Monthly) tables. This table is provided to help you avoid schema issues.

TAR data source	Fields that exist only in Live tables	Fields that exist only in Aggregated tables
ARP cache entries	n/a	n/a
Boot performance	n/a	n/a
Device interaction	InteractionSeconds, LogonSeconds, PresentSeconds	AverageIdleResponsivenessMsCount, AverageInteractiveResponsivenessMsCount, AverageSessionResponsivenessMsCount, InteractionMinutes, LogonMinutes, PresentMinutes

TAR data source	Fields that exist only in Live tables	Fields that exist only in Aggregated tables
Device performance	n/a	SampleCount
Device resource demand	n/a	SampleCount
DNS resolutions	n/a	LookupCount
Operating System performance	n/a	ExecutionCount
Performance event	EventData	EventCount
Process executions	CommandLine, ProcessId, ParentProcessId	ExecutionCount
Process stabilizations	ProcessId, StabilizationTimeMs	ExecutionCount, TotalStabilizationTimeMs
Process usage	n/a	All fields only available in $ProcessUsage_Daily.
Sensitive processes	n/a	DetectionCount
Software installations	IsUninstall	InstallCount, UninstallCount
Software interaction	InteractionSeconds, LogonSeconds, PresentSeconds	AverageIdleResponsivenessMsCount, AverageInteractiveResponsivenessMsCount, AverageSessionResponsivenessMsCount, InteractionMinutes, LogonMinutes, PresentMinutes
Software performance	n/a	SampleCount

TAR data source	Fields that exist only in Live tables	Fields that exist only in Aggregated tables
TCP outbound connections	ProcessId	ConnectionCount
User usage	n/a	All fields only available in $UserUsage_Daily.

1.5.10.4.1 Timestamps

The timestamp column (TS) in each table is stored in Unix Epoch format, defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970.

To convert to a readable text format use the EPOCHTOJSON function. See also datetime handling.

Example - converting Unix Epoch timestamps

```
SELECT Fqdn, EPOCHTOJSON(TS) AS TS_ FROM $DNS_Hourly WHERE Fqdn LIKE "%facebook %";
```

Timestamps are truncated in the aggregated tables.

- Hourly - time is truncated to each hour - so an event that occurred at 2017-01-27 18:03:54 would be included in the summary for **2017-01-27 18:00:00**
- Daily - time is truncated to midnight on each day - so an event that occurred at 2017-01-27 18:03:54 would be included in the summary for **2017-01-27 00:00:00**
- Monthly - time is truncated to midnight on the first day of each month - so an event that occurred at 2017-01-27 18:03:54 would be included in the summary for **2017-01-01 00:00:00**

1.5.10.4.2 ARP cache entries

Windows only. The following table shows fields available in the **$ARP_** tables.

Field	Datatype	Description	Sample value	Tables
CacheCount	integer	The number of times that the combination of IpAddress, MacAddress and Subnet were seen in the ARP cache for this time period.	1234	All
IpAddress	string	The IP address that was resolved using ARP.	192.168.11.12	All
MacAddress	string	The MAC (physical) address to which the IP address was resolved.	58-82-a8-93-4c-da	All

Field	Datatype	Description	Sample value	Tables
Subnet	string	The CIDR-format IP subnet to which the resolved IP address belongs.	192.168.11.0/8	All
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All

The Tachyon client polls the operating system ARP cache periodically. Since the lifetime of an entry in the ARP cache can be variable, if an entry in the ARP cache is encountered which is already present in the Tachyon client's database, the Tachyon client will increment the CacheCount field on the table for that row, and update the timestamp (TS) field to the current time. To that end, the CacheCount field can be used to determine how frequently a particular entry was observed in the operating system's cache.

1.5.10.4.3 Boot performance

Supported on Windows, with support for some features on macOS. The following table shows fields in the **$BootPerformance_Live** table.

Field	Datatype	Supported on	Description	Sample value	Tables
BootAutoChkTimeSeconds	real	Windows	--	0	
BootCriticalServicesInitTimeSeconds	real	Windows	Time in seconds to initialize critical services enumerated in Registry at HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Diagnostics\Performance\Boot	46.38600159	
BootDevicesInitTimeSeconds	real	Windows	Time taken in seconds to identify and initialize devices	0.894999981	
BootDriverInitTimeSeconds	real	Windows	Time taken in seconds to initialize drivers that were loaded by winload.exe	3.105999947	

Field	Datatype	Supported on	Description	Sample value	Tables
BootEndTime	string	Windows, macOS	Timepoint at which the last Boot Event ended	2021-07-01T04:50:53.874Z	
BootExplorerInitTimeSeconds	real	Windows	Time taken by the system to create the desktop window manager (DWM) process, which initializes the desktop and displays it for the first time.	43.48400116	
BootKernelInitTimeSeconds	real	Windows	Time taken by windows kernel to initialize data structures and components.	0.046999998	
BootMachineGroupPolicyProcessingTimeSeconds	real	Windows	Time in seconds taken at boot time to apply Machine Group Policy settings.	1	
BootMachineProfileProcessingTimeSeconds	real	Windows	Time in seconds taken to load and apply machine profile settings at system startup.	0.169	
BootMainPathBootTimeSeconds	real	Windows	Starts when you see the Start Windows splash screen and ends when the desktop appears.	74.66500092	
BootNumStartupApps	integer	Windows	Total number of application marked as startup apps using 'Run' or 'RunOnce' registry keys or placed in the startup folder	8	
BootOsLoaderTimeSeconds	real	Windows	Time taken by Winload.exe to load essential system drivers and initializes the system to the point where the Windows kernel can begin execution.	2.105999947	

Field	Datatype	Supported on	Description	Sample value	Tables
BootOtherLogonInitActivityTimeSeconds	real	Windows	--	0.352999985	
BootPostBootTimeSeconds	real	Windows	Starts when the desktop appears. During this time, services and application may be starting but it is not considered complete until the system has reached a certain idle state.	47.95700073	
BootPrefetchInitTimeSeconds	real	Windows	Time taken by Prefetcher to initialize according to the boot plan.	0	
BootSessionInitStartTimeSeconds	real	Windows	--	4.842999935	
BootStartupAppsTimeSeconds	real	Windows	Time in seconds taken by startup applications to initialize.	18.21299934	
BootStartupDegradationApps	string	Windows	';' delimited list of application which caused boot performance degradation	Antimalware Service Executable:47.254;Cortex XDR Service:26.835	
BootStartupDegradationTimeSeconds	real	Windows	Total degradation time in seconds	74.089	
BootTimeSeconds	real	Windows,	Time in seconds to boot the machine. Is the addition of	122.6220016	

Field	Datatype	Supported on	Description	Sample value	Tables
		macOS	BootMainPathBootTimeSeconds and BootPostBootTimeSeconds		
BootUserGroupPolicyProcessingTimeSeconds	real	Windows	Time in seconds taken at boot time to apply User Group Policy settings.	377	
BootUserLogonWaitTimeSeconds	real	Windows	Time in seconds the OS waited on the logon screen for the user authentication input	24.96800041	
BootUserProfileProcessingTimeSeconds	real	Windows	Time in seconds taken to load and apply user profile settings at system startup.	17.83099937	
BootWinLogonStartTimeSeconds	real	Windows	Time elapsed between the user logon screen appears and the Explorer process starts. The service control manager starts services, and Group Policy scripts run.	12.82499981	
RebootCount	integer	Windows	Number of times the machine has been rebooted on the day of last reboot.	1	
TS	integer	Windows, macOS	When the record was added to the table. See Timestamps.	1625204343	

New in v8.2 is minimal support for boot performance on macOS. (The other boot metrics are very specific to Windows.)

The Tachyon Client polls the Windows Event Logs to search for a new Boot Event (Microsoft-Windows-Diagnostics-Performance/Operational [EventID=100]) . If a newer event different from the one currently stored in the Live table is present, it parses the log to extract the various boot performance metrics. We also poll other event logs (Microsoft-Windows-GroupPolicy/Operational [EventID=8000], [EventID=8001], System / [EventID=1074]) and Startup XML files generated by the OS to gather data for all the above mentioned fields. This information is used by the Tachyon Experience application.

1.5.10.4.4 Device interaction

Windows only. The following table shows fields available in the **$DeviceInteraction_**tables.

Field	Data type	Description	Sample value	Tables
AverageIdleResponsivenessMs	real	The number of milliseconds, on average, that the foreground application took to respond to a probe. A lower value means the foreground application is likely to feel more responsive. This field considers only samples where the user was "idle" - i.e. not interacting with the device using the keyboard/mouse.	0.73310155982222314	All
AverageIdleResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageIdleResponsivenessMs field.	500	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly
AverageInteractiveResponsivenessMs	real	The number of milliseconds, on average, that the foreground application took to respond to a probe. A lower value means the foreground application is likely to feel more responsive.	1.3999095275108098	All

Field	Data type	Description	Sample value	Tables
		This field considers only samples where the user was interacting with the device using the keyboard/mouse.		
AverageInteractiveResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageInteractiveResponsivenessMs field.	500	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly
AverageSessionResponsivenessMs	real	The number of milliseconds, on average, that the foreground application took to respond to a probe. A lower value means the foreground application is likely to feel more responsive. This field considers both idle and interactive samples.	1.0665055436665165	All
AverageSessionResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageSessionResponsivenessMs field.	500	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly

Field	Data type	Description	Sample value	Tables
BusyCursorSeconds	integer	The number of seconds that the user was presented with an hourglass ("busy") cursor.	816	All
InteractionSeconds	integer	The number of seconds that the user was interacting (keyboard and mouse activity) with the device within the aggregated live sample (which corresponds to one minute by default).	8	• $DeviceInteraction_Live
InteractionMinutes	integer	The number of minutes that the user was interacting (keyboard and mouse activity) with the device within the hour, day or month. Note: if a user interacts at all with the device (even a single click/keystroke) in a minute period, that minute is counts towards the total InteractionMinutes value.	316	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly
LogonSeconds	integer	The number of seconds that the user was logged on to the device within the aggregated live sample (which corresponds to one minute by default).	12	• $DeviceInteraction_Live

Field	Data type	Description	Sample value	Tables
LogonMinutes	integer	The number of minutes that the user was logged on to the device within the hour, day or month.	1016	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly
PresentSeconds	integer	The number of second that the user was deemed to be "present" at the device within the aggregated live sample (which corresponds to one minute by default). A user's presence is implied if the user is logged on and the device is not locked.	37	• $DeviceInteraction_Live
PresentMinutes	integer	The number of minutes that the user was deemed to be "present" at the device within the hour, day or month. A user's presence is implied if the user is logged on and the device is not locked.	712	• $DeviceInteraction_Hourly • $DeviceInteraction_Daily • $DeviceInteraction_Monthly
RemoteHost	string	The FQDN (or hostname or IP address, depending on availability) of the device connected remotely to this one,	myotherdevice.acme.local	All

Field	Data type	Description	Sample value	Tables
		e.g. over a Remote Desktop session. An empty value implies a local session - i.e. the user was logged at the console. The Device interaction hourly, daily and monthly tables will aggregate sessions for each distinct combination of UserName and RemoteHost in the given period. So if an individual user connects to this device from three other devices and also from the console in a given hour/day/month, this data will be aggregated into four distinct records.		
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All
UserName	string	The name of the user interacting with the device.	1E\bill.gates	All
UserWaitSeconds	integer	The number of seconds within this sample that the user was deemed to be waiting for the device to respond.	17	All

Field	Data type	Description	Sample value	Tables
		This is the total number of seconds where either the user was presented with an hourglass ("busy") cursor, and/or where the foreground application was taking longer than the defined threshold (25ms by default) to respond to probes. The threshold can be configured using the Module.Inventory.SlowMessageThresholdMs setting. **TODO - link? We're missing this setting?**		

New in 5.1, this capture source is used by the Tachyon Experience application and gets data from the 1E Client UI component (controlled by the Interaction module).

The table contains one row per combination of (period + user + remote (or local) device), and summarizes both user presence/activity and overall responsiveness of applications for that user's session:

- User presence and activity is represented as logon time, present time and interaction time. Note that by definition: logon time >= present time >= interaction time.
- Application responsiveness is measured in milliseconds, and is the time taken for the foreground application to respond to a probe. Separate values are stored depending on whether the user is interacting with the device or is "idle".
- The table also includes data for "busy time" and "wait time". Busy time is when the user is presented with an hourglass cursor; wait time is when the user is either presented with an hourglass cursor OR when the foreground application is slow to respond to a probe.

1.5.10.4.5 Device performance

Supported on Windows, with support for some features on macOS. The following table shows fields available in the **$DevicePerformance_**tables.

Field	Data type	Supported on	Description	Sample value	Tables
DiskOthersAverageQueueLength	real	Windows	Average queue length for non-system disk(s). A high value indicates that these disk(s) are not keeping up with the I/O backlog.	0.041	All
DiskOthersAverageSecondsPerWrite	real	Windows	Average time taken for non-system disk(s) to perform a write. A high value indicates that these disk(s) are taking too long to service write requests.	0.00177	All
DiskOthersFreeSpaceMegabytes	integer	Windows	The free disk space across non-system disk(s). A lower value indicates that these disks are running low on space and space may need to be released.	1513894	All
DiskOthersSplitIoPerSecond	real	Windows	The I/O operations which were broken into multiple requests across non-system disk(s). A high value may indicate excessive disk fragmentation.	0.02669	All
DiskOthersUsageTimePercent	real	Windows	The percentage of time that non-	3.1447	All

Field	Data type	Supported on	Description	Sample value	Tables
			system disk(s) were servicing requests. A high value indicates these disk(s) may be excessively busy.		
DiskSystemAverageQueueLength	real	Windows	Average queue length for the system disk. A high value indicates that this disk is not keeping up with the I/O backlog.	0.06877	All
DiskSystemAverageSecondsPerWrite	real	Windows, macOS	Windows: Average time taken for the system disk to perform a write. A high value indicates that this disk is taking too long to service write requests. macOS: How many disk operations took place over the sample interval, inverted, on all local disks (not just the system disk). In other words, the inverse of (No. of write operations during the sample period) / (Sample duration in seconds)	0.001319 7	All

Field	Data type	Supported on	Description	Sample value	Tables
DiskSystemFreeSpaceMegabytes	integer	Windows, macOS	Windows: The free disk space on the system disk. A lower value indicates that this disk is running low on space and space may need to be released. macOS: The free space on the system disk, measured in megabytes. The system disk is that with the home directory of the (root) user that runs the 1E Client.	27261	All
DiskSystemSplitIoPerSecond	real	Windows, macOS	Windows: The I/O operations that were broken into multiple requests on the system disk. A high value may indicate excessive disk fragmentation. macOS: Not split I/O operations, but the average number of fragmented files over the interval divided by the interval, on all local disks (not just the system disk).	0.9275	All

Field	Data type	Supported on	Description	Sample value	Tables
DiskSystemUsageTimePercent	real	Windows, macOS	Windows: The percentage of time that the system disk was servicing requests. A lower score indicates this disk may be excessively busy. macOS: The percentage of time that all local disks (not just the system disk) spent reading and writing.	3.8339	All
MemoryHardPageFaultsPerSecond	real	Windows, macOS	Windows: The amount of memory pages that had to be read from disk-based storage. A higher value indicates this device is low on available physical memory macOS: The number of pages swapped in per second.	0.2385	All
MemoryPageFileUsagePercent	real	Windows, macOS	The percentage of the page file that is in use. A higher value indicates more high page file use, which may mean the device is	9.6219	All

Field	Data type	Supported on	Description	Sample value	Tables
			low on available physical memory.		
MemoryUsageMegabytes	integer	Windows, macOS	The amount of physical memory in use.	27946	All
MemoryUsagePercent	real	Windows	The percentage of physical memory in use. A higher value indicates higher memory consumption, and therefore less available physical memory.	85.43	All
NetworkActiveTcpConnections	integer	Windows, macOS	The average number of active (inbound and output) TCP connections.	90	All
NetworkBroadcastRate	integer	Windows, macOS	The sum of multicast packets that have been sent and received per second	5	All
NetworkBytesReceivedPerSecond	integer	Windows, macOS	The average number of bytes received per second.	2145	All
NetworkBytesSentPerSecond	integer	Windows, macOS	The average number of bytes sent per second.	1579	All

Field	Data type	Supported on	Description	Sample value	Tables
NetworkNetRetransmitRate	integer	Windows, macOS	The rate of TCP segment (both IPv4 and IPv6) retransmissions done per second.	1	All
NetworkOutputQueueLength	integer	Windows, macOS	The length of the output packet queue (in packets).	0	All
NetworkPacketRate	integer	Windows, macOS	the sum of all TCP packets that have been sent and received per second.	71	All
NetworkPacketsOutboundDiscarded	integer	Windows, macOS	The number of outgoing packets dropped by the network adapter.	1	All
NetworkPacketsOutboundErrors	integer	Windows, macOS	The number of outgoing packets dropped by the network adapter due to errors at physical layer	2	All
NetworkPacketsReceivedDiscarded	integer	Windows, macOS	The number of incoming packets dropped due to non availability of receive buffers to store the incoming frames.	1	All

Field	Data type	Supported on	Description	Sample value	Tables
NetworkPacketsReceivedErrors	integer	Windows, macOS	The number of packets dropped by the network adapter due to various errors at physical layer	4	All
NetworkPacketsReceivedNonUnicastPerSecond	integer	Windows	The rate that non-unicast, that is, subnet broadcast or subnet multicast packets, are delivered to a higher-layer protocol.	7	All
NetworkPacketsSentNonUnicastPerSecond	integer	Windows	The rate that packets are requested to be transmitted to non-unicast, that is, subnet broadcast or subnet multicast, addresses by higher-layer protocols.	3	All
NetworkSaturationPercent	integer	Windows	Not implemented. Deprecated in v8.0 in favour of NetworkUtilizationPercent	0 ; NULL (v8.0 onwards)	All
ActiveUserSampleCount	integer	Windows	Number of samples used to form NetworkUsageAverageUserLoggedIn aggregated data	1	All

Field	Data type	Supported on	Description	Sample value	Tables
NetworkUsageAverageUserLoggedIn	integer	Windows, macOS	Windows: NetworkUtilization Percentage value when the user is active on the machine. macOS: The average network bandwidth used on a device when a user is active on the device. If the user is not logged in then this value is empty. If the user is logged in then it is the same as NetworkUtilization Percent. A "logged in" user is someone using the Mac desktop; SSH sessions are ignored.	23	All
NetworkUtilizationPercent	integer	Windows, macOS	A percentage value that represents the ratio of total network traffic to the reported maximum bandwidth supported by the interface.	25	All
ProcessorInterruptTimePercent	real	Windows	The amount of time the CPU spent servicing interrupts. A higher value may	0.23485	All

Field	Data type	Supported on	Description	Sample value	Tables
			indicate faulty or misconfigured hardware/drivers.		
ProcessorQueueLength	real	Windows	The CPU queue length (backlog of processing work). A higher value means that the CPU is not keeping up with the workload.	0.33	All
ProcessorTimePercent	real	Windows, macOS	The CPU load. A higher value indicates that the CPU is fully loaded and additional processing power may be required.	15.10	All
ProcessorTimeSeconds	real	Windows, macOS	The average number of CPU seconds (amount of processor work) per second. A value of 1 indicates that a single CPU core was entirely busy for a second.	143.12	All
TCPv4ConnectionsEstablished	integer	Windows	The number of TCPv4 connections established by the endpoint	287	All
TCPv6ConnectionsEstablished	integer	Windows	The number of TCPv6 connections established by the endpoint	12	All

Field	Data type	Supported on	Description	Sample value	Tables
SampleCount	integer	Windows, macOS	Number of samples used to form this aggregated data	4244	• $DevicePerformance_Hourly • $DevicePerformance_Daily • $DevicePerformance_Monthly
TS	integer	Windows, macOS	When the record was added to the table. See Timestamps.	1500756083	All
UserRating	integer	Windows	Not yet implemented.	0	All
WirelessReceiveRate	integer	Windows, macOS	The maximum possible link layer receive transfer speed that can be expected from the Wireless networking on the endpoint.	130000	All
WirelessSignalQuality	integer	Windows, macOS	A percentage value that represents the signal quality of the wireless network capability on the endpoint.	80	All
WirelessTransmitRate	integer	Windows, macOS	The maximum possible link layer send transfer speed that can be	71500	All

Field	Data type	Supported on	Description	Sample value	Tables
			expected from the Wireless networking on the endpoint.		
WirelessSampleCount	integer	Windows, macOS	Number of samples used to form Wireless aggregated data	1	All

New in 5.0, this capture source is used by the Tachyon Experience application.

Support for macOS added in v8.2, although not all metrics are collected. (The schemas are the same for Windows and macOS, but the "missing" metrics are null on macOS.)

The Network Metrics collect data for the Primary Network Interface.

1.5.10.4.6 Device resource demand

Windows only. The following table shows fields available in the **$DeviceResourceDemand_**tables.

Field	Datatype	Description	Sample value	Tables
AllocatedMB	integer	How much memory, in megabytes, was allocated to this device on average during this sample.	8192	All
AllocatedMips	integer	How much CPU resource, in Mips (millions of instructions per	34848	All

Field	Datatype	Description	Sample value	Tables
		second), was allocated to this device on average during this sample.		
CpuMips	integer	How much CPU resource in Mips was being used by this device on average during this sample.	1660	All
DiskKBps	integer	How much disk throughput (in kilobytes per second) was used by this device on average during this sample. Throughput is measured across all fixed disks.	739	All

Field	Datatype	Description	Sample value	Tables
MemoryMB	integer	How much memory, in megabytes, was used by this device on average during this sample.	5883	All
NetKBps	integer	How much network throughput (in kilobits per second) was used by this device on average during this sample. Throughput is measured across all network adapters.	121	All
Rttms	integer	The roundtrip time, in milliseconds, for user input to be processed by the virtualization	3	All

Field	Datatype	Description	Sample value	Tables
		infrastructure. Note that only Citrix virtualization technology is supposed when determining the roundtrip time.		
SampleCount	integer	Number of samples used to form this aggregated data	123	• $DeviceResourceDemand_Hourly • $DeviceResourceDemand_Daily • $DeviceResourceDemand_Monthly
Server	string	Where available, the FQDN (or hostname or IP address) of the virtual server that is hosting this virtual machine. If the device is not virtualized, or if the virtual	myvirtualserver.acme.local	All

Field	Datatype	Description	Sample value	Tables
		server information is not available, this field will be empty. Note that only Hyper-V and Citrix virtualization technologies are supported when determining the name of the virtual server.		
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All

New in 5.1, this capture source is used by the Tachyon Experience application.

This data is particularly useful when it has been collected virtual machines - it allows you to compare allocated resources with actual resource demand. This can give insight into whether virtual resources are under- or over-provisioned.

1.5.10.4.7 DNS resolutions

Windows only. The following table shows fields available in the **$DNS_**tables.

Field	Datatype	Description	Sample value	Tables
Fqdn	string	The FQDN which is being resolved.	client-office365-tas.msedge.net	All
LookupCount	integer	Sum of resolutions per FQDN within the hour, day, month.	1234	• $DNS_Hourly • $DNS_Daily • $DNS_Monthly
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All

When using polling, the local DNS cache is queried for all unique FQDNs. This includes an initial scan of cache entries created before the Tachyon client starts, which are stored with the same timestamp. New entries that appear in the cache are deemed to correspond to new resolutions and stored with the timestamp of when the polling occurred.

When using ETW, the Tachyon client attempts to capture DNS queries at the point that they are made. The query is captured, not the result of that query. That is, the Tachyon client will capture a request to resolve an FQDN which may ultimately not be resolvable. The DNS cache is not scanned.

1.5.10.4.8 Operating System performance

Windows only. The following table shows fields available in the **$OperatingSystemPerformance_**tables.

Field	Datatype	Description	Sample value	Tables
CpuSeconds	real	The time taken in seconds (on average) to run the test for the corresponding metric.	9.1E-05	All
Execution Count	integer	The number of times that the test for this metric was run within the hour, day, month.	1	• $OperatingSystemPerformance_Hourly • $OperatingSystemPerformance_Daily • $OperatingSystemPerformance_Monthly

Field	Datatype	Description	Sample value	Tables
Metric	string	A row for each of the following 15 metrics: Metric: BootTime — Description: Most recent time taken in seconds for the device to boot to the logon prompt. A long boot up duration may influence user satisfaction .. Metric: CreateFile — Description: Time taken to create an empty, temporary file. A high value indicates that the operating system may be underperforming for basic file operations. Metric: CreateProcess — Description: The time taken to create a	All	

Field	Datatype	Description	Sample value	Tables
		new process. A high value indicates that the operating system is taking a long time to create new processes.		
	CreateThread	The time taken to create a new thread within a process. A high value indicates that the operating system is taking a long time to create new threads.		
	CreateWindow	The time taken to generate an empty window. A high value indicates poor Windows desktop performanc		

Field	Datatype	Description	Sample value	Tables
		e, which may cause applications to appear unresponsive.		
	DiskRand	The time taken to perform a random access disk operation on the system drive. A high value indicates poor random disk access performance.		
	DiskSeq	The time taken to perform a sequential access disk operation on the system drive. A high value indicates poor sequential disk access performance.		

Field	Datatype	Description		Sample value	Tables
		LoadDLL	The time taken to load and unload a DLL in a process. A high value indicates library loading is slow, which may affect the startup time of applications.		
		Memory	The time taken to allocate, zero and free a block of memory. A high value indicates that the operating system is slow to serve memory requests, which may affect application performance.		

Field	Datatype	Description	Sample value	Tables
		MessageDispatch	The time taken to dispatch and confirm processing of a windows message. A high value indicates poor message processing throughput, which may affect application responsiveness.	
		OpenHandle	The time taken to acquire a basic operating system resource. A high value indicates that applications may underperform because the operating system is slow to service	

Field	Datatype	Description	Sample value	Tables
		resource requests.		
	RegReadHKLM	The time taken to read from the HKLM Windows Registry hive. A high value indicates poor registry read performance, which in turn may affect application performance.		
	RegWriteHKCU	The time taken to write to the HKCU Windows Registry hive. A high value indicates poor registry write performance, which in turn may affect application		

Field	Datatype	Description	Sample value	Tables
		performance.		
	RegWriteHKLM	The time taken to write to the HKLM Windows Registry hive. A high value indicates poor registry write performance, which in turn may affect application performance.		
	UDPSend	The time taken to perform a basic loopback UDP send. A high value may indicate problems with the operating system network stack		

Field	Datatype	Description	Sample value	Tables
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All

New in 5.0, this capture source is used by the Tachyon Experience application.

The Operating System Performance tables store the average time taken to run various tests against the operating system. These tests measure Operating System performance by timing common OS-level tasks such as creating processes and threads, reading and writing to the registry, etc.

1.5.10.4.9 Performance event

Supported on Windows, with support for some features on macOS. The following table shows fields available in the **$PerformanceEvent_**tables.

Field	Datatype	Description	Sample value	Tables
EventData	string	JSON-formatted data, specific to the EventType, containing additional information about this event.	(json data)	• $PerformanceEvent_Live
EventCount	integer	A count of the number of events of this EventType and ReferenceItem combination for the hour, day, or month.	123	• $PerformanceEvent_Hourly • $PerformanceEvent_Daily • $PerformanceEvent_Monthly
EventType	string	The type of event, in dotted-namespace notation. Events captured are:	Software.Install	All

Field	Datatype	Description				Sample value	Tables
		EventType	Supported on	Description	ReferenceItem		
		Device.PerformanceAnomaly	Windows	Reserved for future use	N/A		
		OperatingSystem.Boot	Windows	Occurs when the operating system starts up	N/A		
		OperatingSystem.Crash	Windows, macOS	Windows: Occurs when the operating system crashes (BSOD). macOS: Unexpected shutd			

Field	Datatype	Description			Sample value	Tables
				own of the OS.		
		OperatingSystem.PerformanceAnomaly	Windows	Reserved for future use	N/A	
		OperatingSystem.ServiceFailure	Windows	Occurs when a service fails to start	Service description	
		OperatingSystem.Upgrade	Windows	Occurs when the version number of the operating system changes	N/A	
		Patch.Install	Windows	Occurs when a patch is	N/A	

Field	Datatype	Description			Sample value	Tables
				installed		
		Patch.Uninstall	Windows	Occurs when a patch is removed	N/A	
		Software.Crash	Windows, macOS	Occurs when a process crashes	Executable path of faulted process	
		Software.Hang	Windows	Occurs when an application becomes unresponsive	Executable path of faulted process	
		Software.Install	Windows	Occurs when an application is	Product name	

Field	Datatype	Description				Sample value	Tables
				installed			
		Software.PerformanceAnomaly	Windows	Reserved for future use	N/A		
		Software.Uninstall	Windows	Occurs when an application is removed	Product name		
ReferenceItem	string	A simplified representation, where applicable, of the object to which the event pertains. See the EventType field description for more details.				Microsoft Edge WebView2 Runtime	All
TS	integer	When the record was added to the table. See Timestamps.				1500756083	All

New in 5.0, this capture source is used by the Tachyon Experience application.

From v8.2 some of these events are also reported for macOS.

The data in this table is used to derive a count-over-time of events which may be of relevance when diagnosing performance or end-user experience issues.

1.5.10.4.10Process executions

All features supported on all platforms. The following table shows fields available in the **$Process_**tables.

Field	Data type	Description	Sample value	Tables
CommandLine	string	The full command-line of the process, including (on Windows) the executable name. Sometimes the executable name part of the command-line is quoted, sometimes it's not - it's arbitrary based however the parent process launched the child; so you may see a mix of command-lines like... • "C:\Program Files (x86)\Microsoft Office\root\Office16\OUTLOOK.EXE" • \??\C:\Windows\system32\conhost.exe 0x4 • C:\Windows\system32\svchost.exe -k UnistackSvcGroup	"C:\Windows\system32\VmConnect.exe" "1EUKDEVWKS1231" "TCH-CLI-WXPX86" -G "B2C72520-BBC6-4736-BBBC-5CCF50FE6666" -C "0"	• $Process_Live
Executable Hash	string	The MD5 hash of the process executable.	dae0bb0a7b2041115cfd9b27d73e0391	All
Executable Name	string	The filename (including extension) of the process executable.	vmconnect.exe	All

Field	Data type	Description	Sample value	Tables
ExecutablePath	string	The path and filename of the process executable. On Windows, this is the NT-device format version of the path (as a process does not necessarily need to have been launched from a device which has a drive-letter mapping).	\device\harddiskvolume8\windows\system32\vmconnect.exe	All
ExecutionCount	integer	Sum of executions per executable within the hour, day, month.	1234	• $Process_Hourly • $Process_Daily • $Process_Monthly
ParentExecutableName	string	The filename (including extension) of the executable of the process which spawned this one.	mmc.exe	All
ParentProcessId	integer	The process ID of the process which spawned this one.	2088	• $Process_Live
ProcessId	integer	Operating-system dependent process ID.	178	• $Process_Live

Field	Data type	Description	Sample value	Tables
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All
UserName	string	The name of the user in whose session the process was launched (or blank if it is a system-launched process)	1E\bill.gates	All

On Windows, the 1E Client service runs as LOCAL SYSTEM, therefore details of almost every process will be available to the Tachyon client features; however some processes may not be accessible because of permissions.

The Tachyon client captures process starts; it does not track how long the process has been running, or how much CPU-time (or user/kernel/active time) the process has used.

Each time the Tachyon client starts it does an initial scan of processes before it starts capturing, and will use that time to record when these processes started.

The UserName field is derived from the session in which the process was executed, and doesn't necessarily reflect the user in whose context the process was executed.

1.5.10.4.11Process stabilizations

Windows only. The following table shows fields available in the **$ProcessStabilization_** tables.

Field	Datatype	Description	Sample value	Tables
ExecutableName	string	The filename (including extension) of the process executable.	vmconnect.exe	All
ExecutionCount	integer	Sum of executions per executable and	53	• $ProcessStabilization_Hourly • $ProcessStabilization_Daily

Field	Datatype	Description	Sample value	Tables
		username within the hour, day, month. For example, vmconnect.exe run by 1e\user1 and vmconnect.exe run by 1e\user2 will have separate rows and thus will be summed separately.		• $ProcessStabilization_Monthly
ProcessId	integer	Operating-system dependent process ID.	178	• $ProcessStabilization_Live
StabilizationTimeMs	integer	The time taken for the process to be considered stable, measured in milliseconds. This will be a multiple of 100.	4500	• $ProcessStabilization_Live
TotalStabilizationTimeMs	integer	Sum of the time taken to be considered stable per	864300	• $ProcessStabilization_Hourly • $ProcessStabilization_Daily

Field	Datatype	Description	Sample value	Tables
		executable and username within the hour, day, month. For example, vmconnect.exe run by 1e\user1 and vmconnect.exe run by 1e\user2 will have separate rows and thus will be summed separately.		• $ProcessStabilization_Monthly
TS	integer	When the record was added to the table. See Timestamps .	1500756083	All
UserName	string	The name of the user in whose session the process was launched (or blank if it is a system-launched process)	1e\bill.gates	All

On Windows, the 1E Client service runs as *LOCAL SYSTEM*, therefore details of almost every process will be available; however some processes may not be accessible because of permissions. The Tachyon client captures

only information that can be accessed by *LOCAL SYSTEM* - as such it does not check the UI responsiveness of a process.

By default, process stabilization monitoring is not active. To enable, the process names must be specified in the 1E Client configuration file as follows:

- Add *Module.Inventory.ProcessStabilization.MonitoredProcesses=<string>* to the 1E Client configuration file.
- This is a list of comma separated values, and the case is not significant. For example, *winword.exe* and *WINWORD.EXE* are treated the same.
- The list of monitored processes does not currently have a limit, however adding a large list of processes to monitor can cause performance degradation and the process stabilization time will become less accurate.

Resource usage for a process is tracked, and the process will be considered stable once its resource utilisation has stopped fluctuating. The margin in which a process is considered stable can be modified in the 1E Client configuration file. Changing from default is **not** recommended.

- This margin is controlled by the *fuzziness* configuration setting.
- Add *Module.Inventory.ProcessStabilization.Fuzziness=<integer>* to the 1E Client configuration file. It cannot be lower than 1, and cannot exceed 66. The default is 5.

A process that exits before it is considered stable is not be recorded. Currently, such processes are discarded. A warning is logged when this occurs.

The accuracy of process monitoring decreases if more processes need to be monitored concurrently. For example, accuracy will decrease if many processes are started at the same time. Warnings are logged when this occurs.

The accuracy of the process monitoring decreases if the system is under considerable load, for example high disk or CPU stress.

Aggregation is grouped by the *UserName* and *ExecutableName* fields. Unlike process executions, process stabilization values for *UserName* and *ExecutableName* are lower case.

1.5.10.4.12 Process usage

Windows only. The following table shows fields available in the **$ProcessUsage_Daily** table.

Field	Datatype	Description	Sample value	Tables
CommandLine	string	This is a single instance of the command used to launch that instance, most probably the	C:\Program Files\Git\mingw64\libexec\git-core\git-credential-manager.exe	

Field	Datatype	Description	Sample value	Tables
		first one. It will not contain any differences if other instances are launched with a slightly different comand line. It is an indication of a typical command line for this instance.		
Duration	integer	The number of minutes covered by the individual execution(s) of at least one instance of this executable. Duration can never be more than 1440 minutes, being the number of minutes in a day.	1	
ExecutableHash	string	The MD5 hash of the binary that contains the entry point (usually an exe)	ad3ec70ae9e82582bdf6aa6fd5811376	

Field	Datatype	Description	Sample value	Tables
ExecutableName	string	The name of the binary that contains the entry point obtained from stamped version information where possible, the filename if not.	git-credential-manager.exe	
ExecutableSize	integer	The size of the binary that is hashed below	131168	
ExecutableVersion	string	The version information stamped into the executable where available.	1.5.0.0	
ExecutionCount	integer	The number of instances observed during the Duration period	2	
IsOSProcess	integer	A value of 1 indicates that this is categorised as an operating system by the rules in place.	0	

Field	Datatype	Description	Sample value	Tables
		A value of 0 indicates that it is not.		
LastSeen	integer	The UTC Timestamp of when the last instance of the executable (of all the accumulated subjects of this record) was last seen (polling) or actually exited (events). Whilst any instance is running, for the current day records, LastSeen will creep across the day and duration will increase as time passes if the process remains running. Once midnight is crossed then the daily records for yesterday are 'closed off' by setting	1526982245	

Field	Datatype	Description	Sample value	Tables
		LastSeen = TS + 86400 (the number of seconds in a day), which is midnight of the next day. If all instances of one binary are exited and never run again that day, then the LastSeen field for that daily record should 'stick' at one value and never ever change again. In other words the maximum difference between TS and LastSeen in a single row is at most 86400, being the number of seconds in a day. Tracking of an execution summary from one day to another ("carry-over") can be achieved by		

Field	Datatype	Description	Sample value	Tables
		looking for a record based on $TS_{tomorrow}$ = $LastSeen_{today}$ with all the other key information the same. If that exact key record with the 'carry over' conditions is not found then the process did not theoretically continue across midnight. Note that a process that dies after 23:59:00 and starts before 00:01:00 the next day will appear to be a continuous process in the summary tables. Even though it could theoretically have stopped for nearly two minutes. This is because the resolution of		

Field	Datatype	Description	Sample value	Tables
		the table is to the start of the minute the event occurred in.		
TS	integer	When the record was added to the table. See Timestamps . Midnight UTC that is the start day of the 24 hours covered by this record.	1526947200	

The Tachyon client captures executable usage; this is from the moment the executable is turned into a process, hence the process usage. The Process Usage data presented is grouped by executable binary, and parallel runs are accumulated in the ExecutionCount, but not in the Duration, where coverage time period is desired instead.

1.5.10.4.13 Sensitive processes

A "sensitive process" is one flagged by the Tachyon Performance Metrics program as one which consumes extra CPU when files and processes are created, registry entries are read, etc., suggesting such processes are monitoring such low level O/S operations. Antivirus and other security software legitimately does this (as does for example Windows Explorer and the 1E Client itself), but other processes that do it *may* be a security hazard.

Windows only. The following table shows fields available in the **$SensitiveProcess_**tables.

Field	Datatype	Description	Sample value	Tables
CpuSeconds	real	Average CPU used by the process executable during the	0.456	All

Field	Datatype	Description	Sample value	Tables
		sample intervals.		
DetectionCount	integer	Sum of the number of samples within the hour, day, month in which the process executable was detected.	1	• $SensitiveProcess_Hourly • $SensitiveProcess_Daily • $SensitiveProcess_Monthly
ExecutablePath	string	The path and filename of the process executable. On Windows, this is the NT-device format version of the path (as a process does not necessarily need to have been launched	c:\windows\system32\conhost.exe	All

Field	Datatype	Description	Sample value	Tables
		from a device which has a drive-letter mapping).		
Product	string	The title of the software product.	Microsoft® Windows® Operating System	All
TS	integer	When the record was added to the table. See Timestamps .	1500756083	All
Version	string	Version of the process executable.	10.0.17763.404	All

New in 5.0 this capture source is used by the Tachyon Experience application.

> **Note**
>
> For Windows XP, permissions restrictions mean that not all sensitive processes are detected.

1.5.10.4.14Software installations

All platforms. The following table shows fields available in the **$Software_**tables.

Field	Datatype	Description	Sample value	Tables
Architecture	string	The platform architecture of the software product.	x64	All
InstallCount	integer	Sum of installs per software product version within the hour, day, month. 0 if uninstalled, or present but not detected as installed.	1234	• $Software_Hourly • $Software_Daily • $Software_Monthly
IsUninstall	integer	0 = install, 1 = uninstall.	0	• $Software_Live
Product	string	The title of the software product that was installed/uninstalled.	Google Chrome	All
Publisher	string	The publisher of the software product that was installed/uninstalled.	Google Inc.	All
TS	integer	When the record was added to the table. See Timestamps . The Tachyon client assumes a "new" installation/uninstallation occurred at the point of polling.	1500756083	All
UninstallCount	integer	Sum of uninstalls per software product version within the hour, day, month.	1233	• $Software_Hourly • $Software_Daily • $Software_Monthly

Field	Datatype	Description	Sample value	Tables
		0 if installed, or present but not detected as installed.		
Version	string	The version of the software that was installed/uninstalled.	55.0.2883.87	All

Each time the Tachyon client starts, it does an initial scan of installed software before it starts capturing. Since the Tachyon client has no way of knowing when this install/uninstall happened, it will mark the event as having occurred "now".

On Windows, software installations are read from the registry key *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall* and *HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall*

Per-user installations are not yet supported.

Linux does not distinguish between O/S packages (even the kernel) and application packages; they are all software.

1.5.10.4.15 Software interaction

Windows only. The following table shows fields available in the **$SoftwareInteraction_** tables.

Field	Data type	Description	Sample value	Tables
AverageIdleResponsivenessMs	real	The number of milliseconds, on average, that this application took to respond to a probe when it was in the foreground. A lower value means the application is likely to feel more responsive. This field considers only samples where the user was "idle" - i.e. not interacting with the	0.733101559 82222314	All

Field	Data type	Description	Sample value	Tables
		device using the keyboard/mouse.		
AverageIdleResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageIdleResponsivenessMs field.	500	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily • $SoftwareInteraction_Monthly
AverageInteractiveResponsivenessMs	real	The number of milliseconds, on average, that this application took to respond to a probe when it was in the foreground. A lower value means the application is likely to feel more responsive. This field considers only samples where the user was interacting with the device using the keyboard/mouse.	1.3999095275108098	All
AverageInteractiveResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageInteractiveResponsivenessMs field.	500	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily • $SoftwareInteraction_Monthly

Field	Data type	Description	Sample value	Tables
AverageSessionResponsivenessMs	real	The number of milliseconds, on average, that this application took to respond to a probe when it was in the foreground. A lower value means the foreground application is likely to feel more responsive. This field considers both idle and interactive samples.	1.0665055436665165	All
AverageSessionResponsivenessMsCount	integer	How many aggregated samples were used to derive the value for AverageSessionResponsivenessMs field.	500	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily • $SoftwareInteraction_Monthly
BusyCursorSeconds	integer	The number of seconds that the user was presented with an hourglass ("busy") cursor while this application was in the foreground.	816	All
InteractionSeconds	integer	The number of seconds that the user was interacting (keyboard and mouse activity) with this application within the aggregated	8	• $SoftwareInteraction_Live

Field	Data type	Description	Sample value	Tables
		live sample (which corresponds to one minute by default).		
InteractionMinutes	integer	The number of minutes that the user was interacting (keyboard and mouse activity) with this application within the hour, day or month. Note: if a user interacts at all with the device (even a single click/keystroke) in a minute period, that minute is counts towards the total InteractionMinutes value.	316	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily • $SoftwareInteraction_Monthly
LogonSeconds	integer	The number of seconds that the user was logged on to the device within the aggregated live sample (which corresponds to one minute by default) with this application in the foreground.	12	• $SoftwareInteraction_Live
LogonMinutes	integer	The number of minutes that the user was logged on to the device within the hour, day or month with this application in the foreground.	1016	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily

Field	Data type	Description	Sample value	Tables
				• $SoftwareInteraction_Monthly
PresentSeconds	integer	The number of second that the user was deemed to be "present" at the device within the aggregated live sample (which corresponds to one minute by default) with this application in the foreground. A user's presence is implied if the user is logged on and the device is not locked.	37	• $SoftwareInteraction_Live
PresentMinutes	integer	The number of minutes that the user was deemed to be "present" at the device within the hour, day or month with this application in the foreground. A user's presence is implied if the user is logged on and the device is not locked.	781	• $SoftwareInteraction_Hourly • $SoftwareInteraction_Daily • $SoftwareInteraction_Monthly
ProcessName	string	The name of the foreground process (without the .exe extension) to which this record pertains.	chrome	All
RemoteHost	integer	The FQDN (or hostname or IP address, depending on	myotherdevice.acme.local	All

Field	Data type	Description	Sample value	Tables
		availability) of the device connected remotely to this one, e.g. over a Remote Desktop session. An empty value implies a local session - i.e. the user was logged at the console. The Software interaction hourly, daily and monthly tables will aggregate sessions for each distinct combination of UserName, ProcessName and RemoteHost in the given period. So if an individual user connects to this device from three other devices and also from the console in a given hour/day/month, and has this process in the foreground, this data will be aggregated into four distinct records.		
TS	integer	When the record was added to the table. See Timestamps.	1500756083	All
UserName	string	The name of the user interacting with this application.	1E\bill.gates	All

Field	Data type	Description	Sample value	Tables
UserWaitSeconds	integer	The number of seconds within this sample that the user was deemed to be waiting for the device to respond while this application was in the foreground. This is the total number of seconds where either the user was presented with an hourglass ("busy") cursor, and/or where the foreground application was taking longer than the defined threshold (25ms by default) to respond to probes. The threshold can be configured using the Module.Inventory.SlowMessageThresholdMs setting.	17	All

New in 5.1, this capture source is used by the Tachyon Experience application and gets data from the 1E Client UI component (controlled by the Interaction module).

The table contains one row per combination of (period + user + process name + remote (or local) device), and summarizes both user presence/activity and overall responsiveness for each application which enters the foreground for that user's session:

- User presence and activity is represented as logon time, present time and interaction time. Note that by definition: logon time >= present time >= interaction time.
- Application responsiveness is measured in milliseconds, and is the time taken for the foreground application to respond to a probe. Separate values are stored depending on whether the user is interacting with the application or is "idle".
- The table also includes data for "busy time" and "wait time". Busy time is when the user is presented with an hourglass cursor; wait time is when the user is either presented with an hourglass cursor OR when the application is slow to respond to a probe.

1.5.10.4.16Software performance

Supported on Windows, with support for some features on macOS. The following table shows fields available in the **$SoftwarePerformance_**tables.

Field	Datatype	Supported on	Description	Sample value	Tables
ExecutablePath	string	Windows, macOS	The path and filename of the process executable.	c:\windows\explorer.exe	All
Architecture	string	Windows, macOS	Windows: The CPU the binary was built for. macOS: The OsArchitecture of the device.	x64	All
HandleCount	integer	Windows, macOS	Windows: How many open handles the process has. macOS: The average number of file	1234	All

Field	Datatype	Supported on	Description	Sample value	Tables
			descriptors in use by the process.		
InstanceCount	integer	Windows, macOS	How many instances of the process are active at the same time	2	All
IoReadKilobytesPerSecond	integer	Windows, macOS	Windows: kB read by the process per second. macOS: The average number of disk kilobytes read per second for the process. This excludes bytes read from	2	All

Field	Datatype	Supported on	Description	Sample value	Tables
			pipes and sockets, for example.		
IoWriteKilobytesPerSecond	integer	Windows, macOS	Windows: kB written by the process per second. The average number of disk kilobytes written per second for the process. This excludes bytes written to pipes and sockets, for example.	0	All
MemoryUsagePhysicalKilobytes	integer	Windows, macOS	kB used by the process in physical memory	35780	All

Field	Datatype	Supported on	Description	Sample value	Tables
MemoryUsageVirtualKilobytes	integer	Windows, macOS	Windows: kB used by the process in virtual memory. macOS: The amount of resident memory in use by the application.	9496	All
ProcessorTimePercent	real	Windows, macOS	Windows: Percentage of time the processor is running the process. macOS: The average CPU load generated by the process.	1.11080672689126	All

Field	Datatype	Supported on	Description	Sample value	Tables
ProcessorTimeSecondsPerSecond	real	Windows, macOS	Number of processor seconds consumed by this process per second, where a "processor second" is a single processor core which is fully busy for an entire second.	0.0666526784132013	All
DiskReadIops	integer	Windows	Disk read operations done by a process per second	6	All
DiskWriteIops	integer	Windows	Disk write operations done by a process	2	All

Field	Datatype	Supported on	Description	Sample value	Tables
			per second		
NetworkBytesExchangedPerSecond	integer	Windows	Total network bytes (Sent + Received) exchanged by a process per second	1485956	All
NetworkBytesReceivedPerSecond	integer	Windows	Network bytes received by a process per second	1597601	All
NetworkBytesSentPerSecond	integer	Windows	Network bytes sent by a process per second	1897601	All
NetworkTotalDataReceivedMegabytes	integer	Windows	MB received by the process	94	All
NetworkTotalDataSentMegabytes	integer	Windows	MB sent by the process	214	All

Field	Datatype	Supported on	Description	Sample value	Tables
Product	string	Windows, macOS	The title of the software product.	Google Chrome	All
SampleCount	integer	Windows, macOS	How many samples the aggregated data is based on.	1233	• $SoftwarePerformance_Hourly • $SoftwarePerformance_Daily • $SoftwarePerformance_Monthly
NetworkSampleCount	integer	Windows	How many samples the per process Network aggregated data is based on.	3	All
DiskSampleCount	integer	Windows	How many samples the per process Disk i/o aggregated data is based on.	3	All
TS	integer	Windows,	When the record	1500756083	All

Field	Datatype	Supported on	Description	Sample value	Tables
		macOS	was added to the table. See Timestamps.		
Version	string	Windows, macOS	The version of the software product.	55.0.2883.87	All

New in 5.0, this capture source is used by the Tachyon Experience application.

New in 8.0, the capture sources **SoftwarePerformace.DiskUsage** (collecting DiskReadIops, DiskWriteIops) and **SoftwarePerformance.ProcessNetworkUsage** (collecting NetworkBytesExchangedPerSecond, NetworkBytesReceivedPerSecond, NetworkBytesSentPerSecond, NetworkTotalDataReceivedMegabytes, NetworkTotalDataSentMegabytes) do not have Aggregation tables of their own. The software performance data collected by them is included in the **$SoftwarePerformance** tables. These capture sources can be disabled in isolation to one other but if the parent **SoftwarePerformance** capture source is disabled then they are also disabled.

Support for macOS added in v8.2, although not all metrics are collected. (The schemas are the same for Windows and macOS, but the "missing" metrics are null on macOS.)

1.5.10.4.17 TCP outbound connections

Windows, macOS, and Linux. The following table shows fields available in the **$TCP_**tables.

Field	Datatype	Description	Sample value	Tables
ConnectionCount	integer	Sum of connections to an IP Address and Port by a process within the hour, day, month.	123	• $TCP_Hourly • $TCP_Daily • $TCP_Monthly

Field	Datatype	Description	Sample value	Tables
IpAddress	string	The target remote IP address of the connection, either an IPv4 or IPv6 address. Windows support for IPV6 is limited; the Tachyon client will capture the connections, but the format used to represent the target IPV6 may differ slightly depending on the mechanism used, and may be subject to change in future versions of the Windows Tachyon client.	132.245.77.18 [2001:4860:4860::8888]	All
Port	integer	The target remote port of the connection.	443	All
ProcessId	integer	The operating-system	11828	• $TCP_Live

Field	Datatype	Description	Sample value	Tables
		specific identifier of the process which instigated the connection. Not supported for Mac OSX earlier than Mac OSX Lion (10.7).		
ProcessName	string	The executable filename of the process which instigated the connection Connections originated from system-oriented processes are captured as "(system)"	chrome.exe	All
TS	integer	When the record was added to the table. See Timestamps .	1500756083	All

The Tachyon client captures TCP connections, not UDP connections - as UDP is inherently connectionless (each packet sent is effectively a new connection).

Each time the Tachyon client starts it does an initial scan of connections before it starts capturing. A limitation of the Windows API is means that all established TCP connections, whether inbound or outbound, are captured; there is no way to distinguish between the two. This means that it is possible for the Tachyon client to double-capture a connection if that connection was established before the Tachyon client stops monitoring, and still

exists when the Tachyon client starts monitoring again, for example between Tachyon client restarts. Unlike other capture sources, there is no persistent storage setting to prevent double-counting.

The Tachyon client captures initial "connect" requests, not just successful connection establishment. This means that an attempt to perform a connection will be captured, even if that connection does not complete, for example, because of a timeout, or the server-side does not permit the connection.

1.5.10.4.18User usage

Windows only. The following table shows fields available in the **$UserUsage_Daily** table.

Field	Datatype	Description	Sample value	Tables
Duration	integer	The number of minutes covered by the individual user session(s) of at least one instance of this login. Duration can never be more than 1440 minutes, being the number of minutes in a day.	12	• $UserUsage_Daily
Email	string	The email address that is cached in the system for this user. This may not necessarily be the email address to use to contact the user via corporate email.	abrown@acme.org	• $UserUsage_Daily
FirstName	string	The forename that the system has cached for the user.	Alice	• $UserUsage_Daily
LastName	string	The surname that the system has cached for the user	Brown	• $UserUsage_Daily
LastSeen	integer	The UTC Timestamp of when the last instance of the user session (of all the accumulated	1526990846	• $UserUsage_Daily

Field	Datatype	Description	Sample value	Tables
		subjects of this record) was last seen (polling) or actually exited (events), rounded down to the start of the minute in which the event occurred. Whilst any session is in progress, for the current day records, LastSeen will creep across the day and the duration will increase as time passes if the user remains logged in. That is Duration and LastSeen will increase each time you query the table (with at least a minute between queries). Once midnight is crossed then the daily records for yesterday are 'closed off' by setting LastSeen = TS + 86400 (the number of seconds in a day), which is midnight of the next day. If all users sessions for one user are exited and never occur again that day, then the LastSeen field for that daily record should 'stick' at one value and never ever change again.		

Field	Datatype	Description	Sample value	Tables
		In other words the maximum difference between TS and LastSeen in a single row is at most 86400, being the number of seconds in a day. Tracking of a user session summary from one day to another ("carry-over") can be achieved by looking for a record based on $TS_{tomorrow}$ = $LastSeen_{today}$ with all the other key information the same. If that exact key record with the 'carry over' conditions is not found then the user session did not theoretically continue across midnight. Note that a session that exits after 23:59:00 and starts again before 00:01:00 the next day will appear to be a continuous user session in the summary tables. Even though it could theoretically have not existed for nearly two minutes. This is because the resolution of the table is to the		

Field	Datatype	Description	Sample value	Tables
		start of the minute the event occurred in. See Timestamps.		
SID	string	The Windows NT SID of the user.	S-1-5-21-xxx-yyy-zzz	• $UserUsage_Daily
TS	integer	When the record was added to the table. See Timestamps .	1526947200	• $UserUsage_Daily
Username	string	The user account name, with a domain prefix if applicable. For Windows devices not a in a domain, the 'domain' is the local machine name. For non-Windows devices such as Linux there is no domain part.	aliceb acme\AliceBrown	• $UserUsage_Daily

The Tachyon client captures user sessions (usage); this is from the moment the user instigates a login/logout, hence User Usage. The usage data presented is grouped by SID and Username, and parallel login durations are really the coverage of the time period, not the total time for all the individual sessions.

User and administrator accounts are included, either local or remote. System accounts, and accounts used to run services, are excluded.

Constraints of Legacy OS

In this documentation, the following are referred to as legacy OS. 1E does not provide support for the Tachyon client on these OS. This is because Microsoft has withdrawn support for these OS or they are not significantly used by business organizations.

- **Windows XP**
- **Windows Vista**
- **Windows 7**
- **Windows 8.0**
- **Windows Server 2003**
- **Windows Server 2008**
- **Windows Server 2008 R2**
- **Windows Server 2012**

- **Windows Server 2012 R2**

Please contact 1E if you require support for these legacy OS.

If you experience an issue on these OS, then please try replicating the issue on a supported OS.

1.5.11 Tags

Guidance and examples for using tags in instructions.

1.5.11.1 Tag Types

Tachyon supports two kinds of tag that can be used to set properties on Tachyon Agent devices. The Tachyon Agent stores both types of tag in persistent storage. Values are set or deleted using actions and queried using questions. The differences are described below.

Tag type	Description
Device	Also known as **scopable** tags in Agent methods documentation, and previously known as **Coverage** tags in earlier versions of Tachyon - see info panel below. Can also be used when defining the coverage for questions and actions. A Tachyon **Custom Properties Administrator** must define the names and values for Device tags before they can be applied to devices using Device tag actions. See Defining Device Tags below. Once set by an action, the Tachyon Agent reports Device tags back to the System where they are stored in the Tachyon Master database for each device (in the *Device* table), which is how they can be used to define coverage for questions and actions. Instructions which query tag contents, do so by querying the Agent, they do not query the Master database; the database is used only for coverage. The Tachyon Agent regularly reports back its list of tags. The list consists of a separator and **name=value** for each tag plus a final separator at the end of the list. The entire list of **name=value** pairs and their separators must not exceed 511 bytes. Any tags that exceed this limit are not reported back. Only the tags reported back by each Tachyon Agent can be used for coverage. Therefore, the length of tag names and values should be kept short if many different tags are to be used. The maximum length of a tag name is 16 characters, and the maximum length of a tag value is 32.
Freeform	Also known as **non-scopable** tags in Agent methods documentation. Cannot be used when defining the coverage for a question.

Tag type	Description
	Freeform tags can be set with arbitrary names and values that are defined only when running a Freeform tag action. Freeform tags exist purely on the Agent, where they can be set, their values fetched, and checked for existence. The list of freeform tags consists of a separator and **name=value** for each tag, plus a final separator at the end of the list. There is no limit to the number of freeform tags or the length of the list, and no limit to the length of tag name or tag value.

Tag names are case-insensitive. Tag values are not case-sensitive, and case is preserved.

Note

Prior to 8.0 Device tags were called Coverage tags. Device tags behave exactly the same as Coverage tags.

Only the name has been changed, to help distinguish the difference with Software tags. Device tags are *client-side* and managed using SCALE and used for coverage in Explorer. Software tags are *server-side* and used in Software inventory.

From 8.0 onwards, both Device and Software tags can be used in Management Group rules, provided they are defined in Settings → Custom Properties.

In 1E Client 8.0, a new Windows installer property TAGS was introduced, to support setting of Device and Freeform tags during installation. Please refer to Tachyon client settings.

1.5.11.2 Tag Storage

Tags are stored on devices in Agent persistent storage, and as such persist during re-installation and upgrades. They are deleted only if explicitly deleted or the persistent storage is deleted.

Tag names are alphanumeric strings, which are case-insensitive but returned in upper-case.

Tag values are alphanumeric strings, with case preserved. That is, a number would be represented as a string such as "*99*", and can be empty (zero-length string) if the presence or absence of a tag is sufficient.

Device and freeform tags are stored internally as two distinct lists.

Each list uses | as a separator between each NAME=Value and at the start and end of the list. For example: |DEPT=Sales|LOCATION=London|COUNTRY=UK|REGION=EMEA|

|| is an empty tag list.

A tag name must be unique within a list, so that assigning a value to a tag will create the tag if it does not already exist or overwrite the tag's value if it was already there.

Device and freeform tags can share the same names and they are distinct.

1.5.11.3 Tag Methods

The following methods existing in the Agent Tagging module:

- Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value.
- Tagging.Clear — Delete all tags of the specified type.
- Tagging.Count — Get the quantity of tags of the specified type.
- Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed.
- Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present.
- Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type.
- Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

Each method requires the tag type to be specified as either scopable (*true*) or non-scopable (*false*), with default scopable (*true*).

When setting a tag, you will need to cast a non-text value as text.

Example

```
@toset = 99;
@toset = SELECT CAST(Value AS text) AS Value FROM @toset;
Tagging.Set(Name: "EXAMPLETAG", Value: @toset.Value, Scopable: false);
```

1.5.11.4 Defining Device Tags

Users of the Tachyon Explorer UI can target specific devices based on various attributes, including Device tags. A Device tag may only be used to target devices if the tag's name and allowed values are already defined in the Tachyon system **Custom Properties** admin page. Instructions can then be configured to prompt users for selection of defined name and values. Whilst it is possible to set *other* Device tag names and values on Agents, it is best not to do this because they cannot be used for coverage until their names and values are defined in **Custom Properties** admin page. More details on the Device tag workflow and tutorials for administrators are given in Tachyon version documentation, for example using Tachyon 8.1 see: Tagging client devices.

Device Tag is a type of custom property, and is currently the only type of custom property that is configurable in the Custom Properties admin page. A Tachyon **Custom Properties Administrator** must define the names and values for Device tags before they can be applied to devices using Device tag actions.

It is possible for a Device tag to exist on Agents which do not exist as a Custom Property. This may occur because an instruction has set a Device tag without reference to Custom Properties, or a Custom Property has been deleted from the system after being set on Agents. It is good practice to define custom properties in the admin page, and then use the instruction **Set Device tag %tagname% to %tagvalue%** which uses "value picker" parameters that let you select a tag name and value based on defined custom properties. If you set a Device tag on an Agent that does not exist as a custom property, then it will behave like a freeform tag and you

will not be able to use it to set coverage in the Tachyon Explorer. Furthermore, you will use up space in the 511 character length of the Device tag list.

See the following links in the SDK for details about value picker parameters.

- Instruction Definition Reference (ParameterJson)
- Getting Started (Parameter properties)

While on the subject of Device tags that don't exist as defined custom properties. Consider a hypothetical situation where Device tags have been defined, and set on many Agents, and then the Tachyon server hardware fails and the Master database is lost. If the server is rebuilt using a new database, then you would probably want to re-instate the same Device tags. If you did not have these documented, then you could use the *GetAll* method (see Tag Methods) to query all devices for details of all Device tag names, and use this to re-define Device tags. Alternatively you could start over again by establishing new tags, and using the *Clear* method to delete existing tags on devices.

1.5.12 User Defined Persistent Storage Tables

Guidance and examples for the User Defined Persistent Storage feature.

This feature allows SCALE to store and retrieve tables of data. Tachyon Activity Record (TAR) uses its own persistent storage tables, which are read-only to SCALE, whereas user defined storage tables are fully manageable using SCALE. TAR and user defined storage tables are stored in a local, compressed and encrypted persistent store, which persists during a Tachyon client upgrade, uninstall and re-installation, unless specifically deleted.

A normal table created by SCALE is defined as a @variable which goes out of scope (get destroyed) when a block of code completes. Storage tables let you save and load the contents of a variable for use in code which may run later. This means instructions and policy rules can refer to (and modify) data created by other, previously executed instructions or policy rules. This is a powerful capability as it allows SCALE code to be*stateful*.

A simple example might be an instruction or rule which adjusts a stored value (such as a running counter or sum) each time it is executed and a specific condition is detected, such as when a named process executes. More sophisticated uses might include capturing detailed data - for example, a list of files - and comparing this data to a previous execution of an instruction or rule.

Table names:

- case insensitive alphanumeric strings and can be up to 128 characters in length
- proceeded with a keyword to distinguish them from other forms of persistent storage object.

Table contents are in JSON format. Alphanumeric strings present in the table contents have case preserved. That is, a number would be represented as a string such as "*99*", and storage of an empty table content is permitted.

> **Note**
>
> Updating a table may fail if the disk is out of space or has other file-system problems.

Freeform Tags are also stored in persistent storage, but they only store individual name=value pairs instead of variables (tables).

1.5.12.1 Storage Methods

The following methods existing in the Agent Storage module:

- Storage.Check — Tests the existence of a user defined persistent storage table.
- Storage.Delete — Removes an existing user defined persistent storage table.
- Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present.
- Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository.
- Storage.List — Enumerates all user-defined persistent storage tables.
- Storage.Set — Set or change the value of the named persistent storage table.

Each method requires the user defined persistent storage name to be specified.

1.5.12.2 Examples

Example use of User-defined Persistent Storage

```
@query = NativeServices.RunWmiQuery(Namespace:"root\\cimv2", Query:"Select Capt
ion, WorkingSetSize from win32_Process");
@workingsetsize = SELECT * FROM @query ORDER BY WorkingSetSize DESC  LIMIT 10;

@storage = SELECT "WorkingSetSize.Top10" AS Storage;

//@found = Storage.Delete(Name: @storage);
@found = Storage.Get(Name: @storage);
IF (@found)
     @result = SELECT "SUCCESS: " || Storage || " already exists and has been r
e-written." AS Result FROM @storage;
 ELSE
     @result = SELECT "SUCCESS: " || Storage || " does not exist and has been c
reated." AS Result FROM @storage;
ENDIF;

// replace contents of table
Storage.Set(Name: @storage, Value:@workingsetsize);

@found = Storage.Get(Name: @storage);
IF NOT (@found)
    @result = SELECT "ERROR: Failed to write to " || Storage || "." AS Result F
ROM @storage;
ENDIF;
```

```
// return the result of checking and writing table
SELECT * FROM @result;
```

1.6 Functions and Methods Reference

Tables showing the earliest version of the Agent in which a function or method was implemented, and on which OS, with links to full details.

> **Note**
>
> This page is regularly updated and includes details of functions and methods that are planned for future versions, which may be withdrawn before actual release.

Parameters are mandatory unless specified as optional. The default values for a string, int and boolean are the empty string, 0 and false respectively.

Tachyon Agent Methods

- Agent.CheckForUpdates
- Agent.Echo
- Agent.GetConnectionInformation
- Agent.GetDiagnostics
- Agent.GetExtensibility
- Agent.GetInstructionStatistics
- Agent.GetKeyFiles
- Agent.GetSummary
- Agent.Log
- Agent.Sleep
- Agent.Stagger
- ContentDistribution.Stage
- ContentDistribution.StageAndExecute
- Device.GetCriticality
- Device.GetDisks

Tachyon Agent Built-in Functions

- HttpGetFile
- IsUndefined
- Raise

Tachyon Agent Built-in Keywords

- ERROR
- EVALUATE
- FOREACH IN DO DONE
- IF ELSE ENDIF
- NOCONTENT
- NOTIMPLEMENTED
- RETURN
- true false

Tachyon Agent SQL Functions

- BASENAMEFROMPATH
- COMBINEPATHS
- COMPAREVERSIONS
- DATETRUNC
- DIRECTORYFROMPATH
- EPOCHTOJSON
- EXPAND
- HASHSTRING
- IPINRANGE
- IPV4STRTOINT
- NEWID
- REGEXP
- REGEXPCAPTURE

- Device.GetInboundMappedDriveHistory
- Device.GetInboundPrinters
- Device.GetLocation
- Device.GetOutboundMappedDriveHistory
- Device.GetOutboundPrinters
- Device.GetProcessors
- Device.GetSummary
- Device.SetCriticality
- Device.SetLocation
- Discovery.LookupNetBiosNames
- Discovery.ScanPortsOnDevices
- Discovery.SendArpRequest
- FileContent.AppendText
- FileContent.FindAndAppend
- FileContent.FindAndDelete
- FileContent.FindAndReplace
- FileContent.GetTorrentAnnounceUris (deprecated)
- FileContent.GetTorrentDescribedFiles (deprecated)
- FileContent.GetTorrentInfo (deprecated)
- FileContent.WriteText
- FileSystem.CreateTempFile
- FileSystem.DeleteFileByName
- FileSystem.FindDirectoryByName
- REGEXPREPLACE

- FileSystem.FindFileByName
- FileSystem.FindFileBySizeAndHash
- FileSystem.GetDigitalSignature
- FileSystem.GetDirectoryHash
- FileSystem.GetFile
- FileSystem.GetFileByLine
- FileSystem.GetFileDetails
- FileSystem.GetFileHash
- FileSystem.GetFilePermissions
- FileSystem.GetFilesInDirectory
- FileSystem.GetFilesInFolder (deprecated)
- FileSystem.GetHash (deprecated)
- FileSystem.GetVersionInfo
- FileSystem.PathExists
- Http.Post
- Interaction.GetActiveResponses
- Interaction.GetInteractionState
- Interaction.GetResponseHistory
- Interaction.GetSessions
- Interaction.GetWindows
- Interaction.InvalidateResponse
- Interaction.SetDoNotDisturb
- Interaction.ShowAnnouncement
- Interaction.ShowNotification
- Interaction.ShowQuestion

- Interaction.ShowSurvey
- Inventory.GetTables
- Mobile.GetEmailAddress
- Mobile.GetEncryptionStatus
- Mobile.GetLocation
- Mobile.GetMobileSummary
- Mobile.GetPasswordQuality
- Mobile.IsGPSEnabled
- Mobile.IsPrintingSupported
- NativeServices.GetWmiNames
- NativeServices.RegistryDeleteKey
- NativeServices.RegistryDeleteUserKey
- NativeServices.RegistryDeleteUserValues
- NativeServices.RegistryDeleteValue
- NativeServices.RegistryEnumerateKeys
- NativeServices.RegistryEnumerateUserKeys
- NativeServices.RegistryEnumerateUserValues
- NativeServices.RegistryEnumerateValues
- NativeServices.RegistryGetUserValues
- NativeServices.RegistryGetValue
- NativeServices.RegistryKeyExists

- NativeServices.RegistrySetUserValues
- NativeServices.RegistrySetValue
- NativeServices.RegistryUserKeyExists
- NativeServices.RegistryUserValueExists
- NativeServices.RegistryValueExists
- NativeServices.RunCommand
- NativeServices.RunWmiInstanceMethod
- NativeServices.RunWmiQuery
- NativeServices.RunWmiStaticMethod
- Network.GetArpEntries
- Network.GetConnections
- Network.GetDnsEntriesByAddress
- Network.GetIpAddress (deprecated)
- Network.GetIpAddresses
- Network.GetListeningEndpoints
- Network.GetNetworkSummary
- Network.GetWiFiNetworks
- Network.SetAirplaneMode
- Network.SetBlueToothMode
- Network.SetWiFiMode
- Network.Wakeup
- Nomad.CachedPkgStatus

- Nomad.CancelJob
- Nomad.FindBestPkgCache
- Nomad.GetCachedPkgDetails
- Nomad.GetCachePath
- Nomad.GetCmContentLocations
- Nomad.GetVariable
- Nomad.JobsStatus
- Nomad.Pause
- Nomad.PauseJobQueue
- Nomad.Resume
- Nomad.ResumeJobQueue
- Nomad.RunPkgStatusRequest
- Nomad.RunSsdQuery
- Nomad.Status
- Nomad.Stop
- OperatingSystem.ControlService
- OperatingSystem.GetInstalledUpdates
- OperatingSystem.GetProcesses
- OperatingSystem.GetRunningApps
- OperatingSystem.GetServiceInfo
- OperatingSystem.GetSummary
- OperatingSystem.GetUpdatesPendingReboot
- OperatingSystem.GetUpdateSummary

- OperatingSystem.KillProcess
- OperatingSystem.KillProcesses
- Patch.Deploy
- Patch.GenerateFullInventoryEvents
- Patch.GetMetadata
- Patch.GetStatus
- Patch.List
- Patch.Restart
- Patch.TestConnection
- Policy.ForceStatusReport
- Policy.GetEventSubscriptions
- Policy.GetRuleEvaluationHistory
- Policy.GetRules
- Policy.GetStatistics
- Policy.GetSummary
- Policy.Invalidate
- Scripting.Run
- Scripting.RunText
- Security.CheckSimpleIoc
- Security.DeleteCertificates
- Security.GetCertificates
- Security.GetQuarantineStatus
- Security.ModifyQuarantineWhitelist
- Security.QuarantineDevice
- Security.UnquarantineDevice
- Shopping.LaunchWsaWizard

- Shopping.ShowNotification
- Software.GetInstallations
- Software.GetUserInstallations
- Software.Uninstall
- Storage.Check
- Storage.Delete
- Storage.Get
- Storage.GetRemote
- Storage.List
- Storage.Set
- Tagging.Check
- Tagging.Clear
- Tagging.Count
- Tagging.Delete
- Tagging.Get
- Tagging.GetAll
- Tagging.Set
- Users.FindLocalGroupMember
- Users.GetInboundConnectionHistory
- Users.GetLocalGroupMembers
- Users.GetLoggedOnUsers
- Users.GetOutboundConnectionHistory
- Users.GetPrimaryUser
- Utilities.GenerateSequence
- Utilities.JsonFromTable
- Utilities.SplitLines

- Utilities.TableFromJson

Page last updated on: 20 - September - 2022

1.6.1 Tachyon Agent Methods

Alphabetic listing of Agent methods. Under this page are separate pages with full details for each function.

> **Note**
>
> This documentation refers to the Tachyon Agent, which is the old name for Tachyon client features of the 1E Client. In Tachyon 4.1 the Tachyon Agent was renamed as the 1E Client, which also provides client modules for Nomad, PXE Everywhere, Shopping/WSA, and WakeUp. The Nomad client module has some methods available for use in SCALE also documented here.
>
> **Note**
>
> The names of modules and methods, and the names of their parameters, are case-sensitive.

A module library is represented by a file in the Agent installation *Extensibility* folder called *Tachyon.<module>.library.dll* or *Tachyon.<module>.library.so* depending on the O/S.

Please refer to the parent page Functions and Methods Reference to know the earliest version of the Agent in which a method was implemented, and which platforms they are supported.

Modules

Agent module

- Agent.CheckForUpdates— Check whether extensibility (module and provider) updates are available, and if so to download and install them.
- Agent.Echo— Return the specified text.
- Agent.GetConnectionInformation— Returns Switch connection and Background Channel information.
- Agent.GetDiagnostics— Uses 1E Client performance counters to get various stats.
- Agent.GetExtensibility— Get the list of extensibility modules and providers currently loaded by the agent.
- Agent.GetInstructionStatistics— Returns execution statistics for instructions that the agent has processed since the last time it was started/restarted.

- Agent.GetKeyFiles— Returns names and paths of key files relating to the execution of the Tachyon Agent.
- Agent.GetSummary— Returns high-level information about the device the Agent is running on.
- Agent.Log— Write a message to the Tachyon Agent log file.
- Agent.Sleep— Pause for the specified number of seconds.
- Agent.Stagger— Delay for a random amount of time up to a limit.

ContentDistribution module

- ContentDistribution.Stage— Downloads content from a web server.
- ContentDistribution.StageAndExecute— Similar to *Stage* (including optional Nomad integration), downloading content from a web server, but also executes a command on successful download.

Device module

- Device.GetCriticality— Gets the user-defined criticality level for this device.
- Device.GetInboundMappedDriveHistory— Retrieves information about any connected shared folders or drives this device uses on remote servers.
- Device.GetInboundPrinters— Retrieves information about any connected shared printer servers
- Device.GetLocation— Gets the user-defined location for this device.
- Device.GetOutboundMappedDriveHistory— Retrieves information about any shared folders this device is exposing.
- Device.GetOutboundPrinters— Retrieves information about any printers attached physically that might be shared
- Device.GetProcessors— Retrieves information about the CPUs plugged in to the motherboard of the device.
- Device.GetSummary— Retrieves a summary of device information, corresponding to (a subset of) the data that is exchanged as part of the Agent's "online" message.
- Device.SetCriticality— Sets the user-defined criticality level for this device.

- Device.SetLocation— Sets the user-defined location for this device.

Discovery module

- Discovery.LookupNetBiosNames— Performs a NetBios reverse lookup on a list of IP addresses
- Discovery.ScanPortsOnDevices— Probe a set of ports on a set of devices.
- Discovery.SendArpRequest— Performs an ARP request to retrieve the physical address for the supplied IPv4 address.

FileContent module

- FileContent.AppendText— Append text to a file.
- FileContent.FindAndAppend— Appends a string to the end of a file if it does ***not*** contain a specified string.
- FileContent.FindAndDelete— Deletes a string from a file.
- FileContent.FindAndReplace— Replaces a string in a file with another string.
- FileContent.GetTorrentAnnounceUris (deprecated)— Returns all announce URIs contained within the *announce* and *announce-list* structures contained inside a torrent file.
- FileContent.GetTorrentDescribedFiles (deprecated)— Returns all files described within the *info* and *files* structures contained inside a torrent file.
- FileContent.GetTorrentInfo (deprecated)— Returns information contained inside a torrent file.
- FileContent.WriteText— Writes text to a file, overwriting any existing contents.

FileSystem module

- FileSystem.CreateTempFile— Creates an empty file in the OS temporary files location, reserving a unique name, and returns its path.
- FileSystem.DeleteFileByName— Deletes a file.
- FileSystem.FindDirectoryByName— Finds directory paths given a leaf directory name.
- FileSystem.FindFileByName— Finds a file by name.
- FileSystem.FindFileBySizeAndHash— Finds a file by its size and hash.

- FileSystem.GetDigitalSignature— Returns all certificates used in an authenticode signature that are not used for timestamping.
- FileSystem.GetDirectoryHash— Returns the hash of a directory and its entire contents, including the contents of subdirectories recursively.
- FileSystem.GetFile— Returns the content of a text file.
- FileSystem.GetFileByLine— Returns the lines of the given text file(s) matching the supplied regular expression pattern.
- FileSystem.GetFileDetails— Retrieves operating system attributes related to the specified file.
- FileSystem.GetFileHash— Returns the hash of a file.
- FileSystem.GetFilePermissions— Returns the set of permissions for a specified file or directory.
- FileSystem.GetFilesInDirectory— Returns the filename, size, created time and modified time of each file in a directory.
- FileSystem.GetFilesInFolder (deprecated)— Returns the filename, size, created time and modified time of each file in a directory.
- FileSystem.GetHash (deprecated)— Returns the hash of a file or directory.
- FileSystem.GetVersionInfo— Returns version and some other details about a file.
- FileSystem.PathExists— Checks whether a path exists.

Http module

- Http.Post— Send the supplied Body text via a HTTP request to the specified Uri endpoint to consume and return the response if completed. This Method is primarily for use with Splunk Event Collectors, but it might support other similar Web APIs.

Interaction module

- Interaction.GetActiveResponses— List all active survey responses.
- Interaction.GetInteractionState— Display the User Interaction state for all user sessions.
- Interaction.GetResponseHistory— Get a response history of User Interaction surveys.
- Interaction.GetSessions— Get a list of user sessions.

- Interaction.GetWindows— List all open windows within user sessions.
- Interaction.InvalidateResponse— Invalidate a User Interaction survey response.
- Interaction.SetDoNotDisturb— Set or unset User Interaction Do Not Disturb mode for a user session.
- Interaction.ShowAnnouncement— Displays a pop-up announcement to one or more user sessions.
- Interaction.ShowNotification— Displays a pop-up notification to one or more user sessions.
- Interaction.ShowQuestion— Display a pop-up question to one or more user sessions.
- Interaction.ShowSurvey— Display a pop-up survey to one or more user sessions.

Inventory module

- Inventory.GetTables— Returns details about the $tables used by Tachyon Activity Record.
- Inventory.EvaluateItem — Signal extended inventory item for evaluation. It is recommended that this is only run on small number of devices at a time.
- Inventory.GetLocalReplica — Get the local replica table for an extended inventory item if it exists.
- Inventory.GetReplicationAudit — Get the replication audit table for an extended inventory item if it exists.
- Inventory.GetReplicationState — Get the extended inventory replication state table if it exists.

Other methods used by the Tachyon Activity Record feature are not exposed for use in SCALE.

Mobile module

- Mobile.GetEmailAddress— Get the email address registered on this device.
- Mobile.GetEncryptionStatus— Get the data encryption status of this device.
- Mobile.GetLocation— Get the physical location of this device in terms of longitude, latitude and altitude.
- Mobile.GetMobileSummary— Get a summary of the mobile device attributes.
- Mobile.GetPasswordQuality— Get the device's password quality, returning the quality of the worst password on the device.

- Mobile.IsGPSEnabled— Determine the GPS status of the device.
- Mobile.IsPrintingSupported— Determine if direct printing is supported on the device.

NativeServices module

- NativeServices.GetWmiNames— Where they exist, returns all the namespaces, classes and columns that match the locator strings.
- NativeServices.RegistryDeleteKey— Delete a registry key.
- NativeServices.RegistryDeleteUserKey— Deletes a given key and all subkeys and values for every user under HKU.
- NativeServices.RegistryDeleteValue— Delete a registry value.
- NativeServices.RegistryEnumerateKeys— Enumerate the subkeys under a registry hive & key.
- NativeServices.RegistryEnumerateUserKeys— Enumerate the subkeys under a registry key for each user in the HKEY_USERS hive.
- NativeServices.RegistryEnumerateUserValues— Get all registry values under a given key for every user under HKU.
- NativeServices.RegistryEnumerateValues— Enumerate the values under a registry hive and key.
- NativeServices.RegistryGetUserValues— Fetch a registry value for every user under HKU.
- NativeServices.RegistryGetValue— Get a registry value.
- NativeServices.RegistryKeyExists— Test whether a registry key exists.
- NativeServices.RegistrySetUserValues— Sets a registry value for every user under HKU. This will create the value and any key hierarchy required.
- NativeServices.RegistrySetValue— Set a registry value.
- NativeServices.RegistryUserValueExists— For each user under HKU, tests whether a given value exists.
- NativeServices.RegistryValueExists— Test whether a registry value exists.
- NativeServices.RunCommand— Execute a command-line.
- NativeServices.RunWmiInstanceMethod— Run a WMI object method.

- NativeServices.RunWmiQuery— Execure a WMI query.
- NativeServices.RunWmiStaticMethod— Run a WMI class method.

Network module

- Network.GetArpEntries— Get current list of known MAC address to IP address mappings from the network adapter table and the ARP cache.
- Network.GetConnections— Get current established TCP connections, for IPv4 and also IPv6 if available.
- Network.GetDnsEntriesByAddress— Get the contents of the DNS client cache.
- Network.GetIpAddress (deprecated)— Gets IP addresses assigned to the machine.
- Network.GetIpAddresses— Gets IP addresses assigned to the machine.
- Network.GetListeningEndpoints— Get the list of TCP and UDP endpoints which are listening for incoming connections, for IPv4 and also IPv6 if available.
- Network.GetNetworkSummary— Get a summary of the device's network attributes.
- Network.GetWiFiNetworks— Returns a list of visible WiFi networks.
- Network.SetAirplaneMode— Turn airplane mode, also known as flight mode, on or off.
- Network.SetBlueToothMode— Turn Bluetooth on or off.
- Network.SetWiFiMode— Turn WiFi mode on or off.
- Network.Wakeup— Send a Wake-on-LAN (WOL) request to a dormant device.

Nomad module

- Nomad.CachedPkgStatus— Get the percentage of the specified content in the local Nomad cache.
- Nomad.CancelJob— Cancels download job with the given ID and removes it from Nomad’s job queue.
- Nomad.FindBestPkgCache— Find the device that has the greatest percentage of the specified content in its Nomad cache.
- Nomad.GetCachedPkgDetails— Gets details about a package cached in Nomad.
- Nomad.GetCachePath— Gets the path on disk where Nomad’s cache is located.

- Nomad.GetCmContentLocations— Find the Configuration Manager locations (on distribution points - DPs) that have the specified content.
- Nomad.GetVariable— Gets the value of the given Nomad "variable".
- Nomad.JobsStatus— List active Nomad jobs.
- Nomad.Pause— Pause any currently-executing Nomad job. Has no effect if no job is currently running.
- Nomad.PauseJobQueue— Pause the Nomad job queue so that no jobs will run.
- Nomad.Resume— Resume any currently-paused Nomad job. Has no effect if no job is currently paused.
- Nomad.ResumeJobQueue— Resume the currently-paused Nomad job queue. Has no effect if the job queue is not currently paused.
- Nomad.RunPkgStatusRequest— Test whether any other SSD (single site download) device has more content than the local Nomad cache.
- Nomad.RunSsdQuery— Use Nomad's SSD (single site download) feature to find on which devices the specified content is cached.
- Nomad.Status— Get the status and version of the Nomad service.
- Nomad.Stop— Stop any currently-executing Nomad job. Has no effect if no job is currently running.

OperatingSystem module

- OperatingSystem.ControlService— Action to start, stop, etc. a service or daemon.
- OperatingSystem.GetInstalledUpdates— Get list of installed Windows updates.
- OperatingSystem.GetProcesses— Get details of executing processes.
- OperatingSystem.GetRunningApps— Get details of running applications.
- OperatingSystem.GetServiceInfo— Retrieves a list of all services, or a single named service.
- OperatingSystem.GetSummary— Get information about the installed version of the operating system.
- OperatingSystem.GetUpdatesPendingReboot— Get list of installed Windows updates that are awaiting a reboot.

- OperatingSystem.GetUpdateSummary— Get status details of Windows automatic update.
- OperatingSystem.KillProcess— Action to terminate a single running process.
- OperatingSystem.KillProcesses— Terminate a running process (or set of process instances of the same executable).

Patch module

- Patch.Deploy— Deploy (i.e. download and by default install) either all available patches or specific patches to this device.
- Patch.GenerateFullInventoryEvents— Generate Full Inventory Events
- Patch.GetMetadata— Query metadata for a single patch or comma-separated list of patches.
- Patch.GetStatus— Report how updates are configured for the device.
- Patch.List— List the status of patches.
- Patch.Restart— Microsoft-suggested procedure to fix Windows Update Agent problems, which has optional steps that include a restart of the Windows Update service.
- Patch.TestConnection— Check that the source for patches is accessible.

Policy module

- Policy.ForceStatusReport— Force a report of policy rule status the next time the Agent restarts.
- Policy.GetEventSubscriptions— Get the list of event subscriptions in the current policy.
- Policy.GetRuleEvaluationHistory— Get a chronological history of recent rule evaluation (for one rule, or all rules).
- Policy.GetRules— Get the list of rules, along with their status, in the current policy.
- Policy.GetStatistics— Returns a single row with policy-related statistics for the lifetime of the Tachyon Agent process.
- Policy.GetSummary— Get a summary of the current policy applied to this Tachyon Agent.
- Policy.Invalidate— Reset this Agent to have a null/empty policy.

Scripting module

- Scripting.Run— Execute the specified script file.
- Scripting.RunText— Execute the specified script supplied as text.

Security module

- Security.CheckSimpleIoc— Evaluates an IOC (indicator of compromise) definition on a device.
- Security.DeleteCertificates— Deletes every certificate in the local machine cert store (as the agent is run as Local System) with the specified thumbprint.
- Security.GetCertificates— Returns every certificate in the local machine cert store (as the agent is run as Local System).
- Security.GetQuarantineStatus— Gets the current quarantine status of the device.
- Security.ModifyQuarantineWhitelist— Allows connections from additional specified IP addresses to an already-quarantined device. This allows security tools other than Tachyon to diagnose and protect the device.
- Security.QuarantineDevice— Attempts to force the device to only communicate using Tachyon.
- Security.UnquarantineDevice— Allows devices to communicate with devices other than the Tachyon Switch and Background channel servers.

Shopping module

No methods in this module are exposed for use in SCALE.

Software module

- Software.GetInstallations— Gets a list of the installed software available to all users, optionally filtered by product and/or publisher.
- Software.GetUserInstallations— Gets a list of the software installed for specific users, optionally filtered by product and/or publisher.
- Software.Uninstall— Uninstall the specified software item.

Storage module

- Storage.Check— Tests the existence of a user defined persistent storage table.

- Storage.Delete— Removes an existing user defined persistent storage table.
- Storage.Get— Indicate whether a persistent storage table of the specified name is present and return its contents if present.
- Storage.GetRemote— Retrieve the requested datum directly from the Platform central repository.
- Storage.List— Enumerates all user-defined persistent storage tables.
- Storage.Set— Set or change the value of the named persistent storage table.

Tagging module

- Tagging.Check— Tests the existence of a named tag of a specified type, optionally with the specified value.
- Tagging.Clear— Delete all tags of the specified type.
- Tagging.Count— Get the quantity of tags of the specified type.
- Tagging.Delete— Delete the named tag of the specified type, and indicate if it originally existed.
- Tagging.Get— Indicate whether a named tag of the specified type is present and return its value if present.
- Tagging.GetAll— Fetch the tag list (containing all tags) of the specified type.
- Tagging.Set— Set (or change if already set) the named tag of the specified type to the specified value.

Users module

- Users.FindLocalGroupMember— List the local groups of which the specified user is an immediate member.
- Users.GetInboundConnectionHistory— List the historical inbound Terminal Services (RDP) connection history
- Users.GetLocalGroupMembers— List the members of a local group.
- Users.GetLoggedOnUsers— List the active (currently logged-on) user sessions.
- Users.GetOutboundConnectionHistory— List the historical outbound Terminal Services (RDP) connection history

- Users.GetPrimaryUser— Get the primary user of a device.

Utilities module

- Utilities.GenerateSequence— Build a sequence of numeric values, starting at 1 up to and including the value of a specified limit.
- Utilities.JsonFromTable— Serializes a table variable into a JSON string
- Utilities.SplitLines— Convert a delimited fragment of text into multiple rows.
- Utilities.TableFromJson— Deserializes a correctly escaped JSON string that represents a two dimensional set of data with consistent columns and rows into a table variable.

WakeUp module

No methods in this module are exposed for use in SCALE.

1.6.1.1 Agent.CheckForUpdates

Method	**CheckForUpdates**
Module	Agent
Library	Core
Action	Check whether extensibility (module and provider) updates are available, and if so to download and install them.
Parameters	(None)
Return values	*Status* (string): e.g. "*Checking for updates*" *OperatingSystem* (string): e.g. "*Windows*" or "*Linux*" *Architecture* (string): e.g. "*x64*"
Example	`Agent.CheckForUpdates();`
Platforms	• Windows • Linux • MacOS

Method	**CheckForUpdates**
Notes	**Warning** Extensibility is deprecated from version 23.5. The method name is slightly misleading because it does not just check for updates but if there are updates available it also downloads and applies them. Because of the way that updates are applied, the method completes before this happens. Therefore if, for example, download of the updates then fails for some reason, the method appears to succeed. To verify that the updates have been applied, run the *GetExtensibility* method afterwards. If *NomadContentDownloadEnabled* is set to *true* in the Agent's configuration file, this method will attempt to use Nomad if it is running. Nomad v6.1.100 or later is supported because of the need to download files as well as directories. It cannot use v6.0.100 because it can only download directories. Prior versions of Nomad are not supported at all. If Nomad is not available or fails then the Agent falls back to its internal HTTP provider.

1.6.1.2 Agent.Echo

Method	**Echo**
Module	Agent
Library	Core
Action	Return the specified text.
Parameters	*Message* (string): Basis of the text to bounce back. *ReplySize* (int; optional, default 1): Repeat the output text as many times as necessary to make the output this number of characters. *Rows* (int; optional, default 1): Rather than just a single row of output, generate this many.
Return values	*Message* (string): The input *Message* text with "*from hostname*" appended, where of course *hostname* is the hostname.
Example	`Agent.Echo(Message:"The quick brown fox jumps over the lazy dog ", ReplySize: 60000);`

Method	**Echo**
	`Agent.Echo(Message:"The quick brown fox jumps over the lazy dog", Rows: 4000);`
Platforms	• Windows • Linux • MacOS
Notes	Useful for quick testing of responses from all agents.

1.6.1.3 Agent.GetConnectionInformation

Method	**GetConnectionInformation**
Module	Agent
Library	Core
Action	Returns Switch connection and Background Channel information.
Parameters	(None)
Return values	*`Slot`* (int): The Switch slot number on the given Switch worker hosting the Agent connection. *`SwitchConnectionTime`* (string): The UTC date and time that the Agent connected to the Switch. See notes. *`SwitchId`* (int): The database identifier of the Switch instance to which the Agent is connected. *`SwitchName`* (string): The name of the Switch instance to which the Agent is connected. (Not the hostname of the Switch.) *`Worker`* (int): The Switch worker number hosting the Agent connection.

Method	**GetConnectionInformation**
BGCCertThumbprint (string): Thumbprint of the certificate received from the Background Channel. *BGCConnected* (string): Indicates whether the last Background Channel connection succeeded. *BGCConnectionTime* (string). The UTC date and time that the Agent last attempted a Background Channel connection. *ClientCertThumbprint* (string): Thumbprint of the certificate sent to the Switch. *LocalIpAddress* (string): Local IP address that the Agent is using to connect to the Switch. *LocalPort* (string): Local port that the Agent is using to connect to the Switch. *RemoteIpAddress* (string): Remote IP address of the Switch. *RemotePort* (string): Remote port of the Switch. *SwitchCertThumbprint* (string) Thumbprint of the certificate received from the Switch. *SwitchClockSkewSeconds* (int): The number of seconds ahead (negative if behind) of UTC time that this Agent is relative to the Switch based on most recent timestamp exchange (see notes).	
Example	`Agent.GetConnectionInformation();`
Platforms	• Windows

Method	**GetConnectionInformation**
	• Linux • MacOS
Notes	If the Nomad module is enabled then the Agent will not access the Background Channel directly and cannot report on that connection. The device's clock skew relative to the Switch is determined each time it connects to the Switch, and when an acknowledged keep-alive packet is transferred (typically in the order of hours/days). You can therefore use the *SwitchConnectionTime* field returned by this method to determine an upper-bound to the age of this information. If the 1E Client has not yet connected to the Switch since it started, or is connected to an older version of the Switch which does not support timestamp negotiation, *SwitchClockSkewSeconds* will be NULL. **Note** Return values are always strings. If you need to perform comparison or mathematical operations on numbers then you need to CAST the string.

1.6.1.4 Agent.GetDiagnostics

Method	**GetDiagnostics**
Module	Agent
Library	Core
Action	Uses 1E Client performance counters to get various stats.
Parameters	(None)
Return values	*BytesReceived* (int): The number of bytes received by the agent from its Switch and the background channel.

Method	**GetDiagnostics**

BytesSent (int): The number of bytes sent by the agent to its Switch and the background channel.

ConnectionsFailed (int): The number of unsuccessful connections to the Switch since the agent started.

ConnectionsSuccessful (int): The number of successful connections to the Switch since the agent started.

CrashDumpsCollected (int): The current number of crash dumps on the host. Implemented on Windows only; always 0 on other platforms.

ExecutionsErrored (int): The number of unsuccessfully executed instructions that the agent has acted on since the agent started.

ExecutionsNotImplemented (int): The number of (unsuccessful) instructions that the agent could not act on because a module or method is not implemented, since the agent started.

ExecutionsPayloadTooLarge (int): The number of responses that failed when attempting to send them over the background channel, since the agent started.

ExecutionsSuccessful (int): The number of successfully executed instructions that the agent has acted on since the agent started.

StartupTime (string): When the agent last started.

StringPoolEntries (int): The number of entries in the agent string pool (shared string instances).

StringPoolSizeBytes (int): The approximate number of bytes of data contained within the string pool.

StringPoolLookups (int): The number of lookups performed on the string pool.

Example

```
Agent.GetDiagnostics();
```

Platforms

- Windows
- Linux
- MacOS

Notes

1.6.1.5 Agent.Sleep

Module	Agent
Library	Core
Action	Pause for the specified number of seconds.
Parameters	*Seconds* (int): The number of seconds for which to to pause; must be >0. *Force* (boolean; optional, default *false*): Usually, if the 1E Client is shutting down then a sleep in progress will be aborted. However, forcing the sleep causes it to block shutdown. This is only useful to test something that happens during shutdown, it is not recommended to delay shutdown in most situations. *Exclusive* (boolean; optional, default *false*): Usually the sleep will affect only the instruction's worker thread, and other instructions running in parallel on other threads are not blocked. If this flag is set then this worker will wait for any already-set exclusivity to finish, then set the exclusivity block while it runs so that other exclusivity users will be blocked until the sleep has completed.
Return values	*SleepTime* (int): The same as *Seconds*.
Example	`Agent.Sleep(Seconds:120, Force:true, Exclusive:true);`
Platforms	• Windows • Linux • MacOS
Notes	Only useful for testing.

1.6.1.6 Agent.GetExtensibility

Method	**GetExtensibility**
Module	Agent
Library	Core
Action	Get the list of extensibility modules and providers currently loaded by the agent.
Parameters	(None)

Method	**GetExtensibility**
Return values	*Type* (string): "*Module*" or "*Provider*". *Name* (string): The name of the module or provider. *Version* (string): The version of the library (.dll, .so, etc) that contains this module or provider. *BinaryName* (string): The name of the library file that contains this module or provider (e.g. "*Tachyon.Module.Core.dll*").

Example	`Agent.GetExtensibility();`
	The following code tests to see if the Agent version is 4.x and the Patch module is loaded.
	`@ver = Agent.GetSummary();`
	`@ext = Agent.GetExtensibility();`
	`SELECT 1 WHERE (SELECT 1 FROM @ver WHERE @ver.Version LIKE '4.%' AND @ver.OsType LIKE '%Windows%') AND (SELECT 1 FROM @ext WHERE @ext.Name LIKE 'Patch');`
	`evaluate;`

Platforms	• Windows • Linux • MacOS
Notes	**Warning** Extensibility is deprecated from version 23.5. Providers supply low-level functionality, but their interfaces are not documented because they are an older technology that has been superseded by internal services within the 1E Client executable itself. These services' interfaces are exposed in the Tachyon Agent SDK, but the SDK is not supplied to customers.

1.6.1.7 Agent.GetSummary

Method	**GetSummary**
Module	Agent
Library	Core
Action	Returns high-level information about the device the Agent is running on.

Method	**GetSummary**
Parameters	(None)

Return values	*TimeZone* (int): The device's current timezone offset from UTC in minutes (positive or negative).
	
	Note
	This is based on timezone, not UTC bias. In other words, this value will not be affected by daylight savings.
	
	Manufacturer (string): The hardware manufacturer of the device's chassis.
	
	OsType: (string): The type of operating system family running on the device. The values are:
	
	• –
	  • "*Linux*"
	  • "*MacOS*" (N.B. **not** "*macOS*")
	  • "*Windows*"
	
	OsArchitecture: (string): The processor architecture of the operating system. The values depend on the *OsType* as follows:
	
	• –
	  • Linux: "*x86*" (32-bit) or "*x86_64*"; same as the result of the "*uname -m*" command.
	  • MacOS: "*arm64*", "*x86*" (32-bit) or "*x64*".
	  • Windows: "*x86*" (32-bit) or "*x64*".
	
	OsVersionText: (string): The caption of the operating system running on the device - a more descriptive version of *OsType*.
	
	Version: (string): The Agent version number in four-part dotted representation (e.g. "23.1.0.23").
	
	VirtualizationPlatform: (string): The type of virtualization technology on which the device's operating system is running (if any).

Example	`Agent.GetSummary();`

Method	**GetSummary**
Platforms	• Windows • Linux • MacOS
Notes	See also Device.GetSummary

1.6.1.8 Agent.GetInstructionStatistics

Module	Agent
Library	Core
Action	Returns execution statistics for instructions that the agent has processed since the last time it was started/restarted.
Parameters	(None)
Return values	For each instruction for which the agent has data: • *CompressionStatus* (string): One of: – • *Compressed* - the data returned by the agent was compressed to save network bandwidth • *CompressionDisabled* - the agent was configured not to compress data • *CompressionFailed* - an error ocurred during the compression process • *DidNotCompressEnough* - there wasn't enough saving achieved by compression to merit sending the data compressed • *TooSmallToCompress* - the payload did not meet the minimum threshold size to apply compression • *ExecutionTime* (string): Datetime (UTC) when the instruction was executed. • *ExecutionTimeMs* (int): The execution time in milliseconds.

Module	Agent

- *InstructionId* (int): ID of the instruction.
- *IsCompressedPayload* (boolean): Whether the payload was compressed or not.
- *PayloadBytes* (int): The uncompressed size of the payload.
- *PayloadBytesCompressed* (int): The compressed size of the payload.
- *RowCount* (int): The number of rows in the payload.
- *SequenceNumber* (int): The sequence number of the instruction.
- *Status* (int): The summary status code from execution of the instruction:
  -
    - 0 = success (with output content)
    - 1 = success (but no output content)
    - 2 = missing provider; missing or invalid parameter; syntax error
    - 3 = unsupported module or method
    - 4 = response too large
- *ThreadId* (string): Operating System-specific identifier of the thread that executed the instruction.

Example

```
Agent.GetInstructionStatistics();
```

Platforms

- Windows
- Linux
- MacOS

Notes

The agent stores instruction statistics in memory, so statistics will only be available for instructions which were executed since the agent last started.

1.6.1.9 Agent.Stagger

Method	**Stagger**
Module	Agent

Method	**Stagger**
Library	Core
Action	Delay for a random amount of time up to a limit.
Parameters	*Limit* (int): The maximum number of seconds for which to to delay. If the value is less than 1 then the value of **DefaultStaggerRangeSecs** in the agent configuration file is used.
Return values	*StaggerTime* (int): The actual delay in milliseconds.
Example	`Agent.Stagger(Limit:30);`
Platforms	• Windows • Linux • MacOS
Notes	Useful for ensuring that multiple agents do not all execute following instructions at the same time, to prevent overloading a shared resource. If the value is less than 1 then**DefaultStaggerRangeSecs**is used. This includes 0, therefore the mimimum delay is 1 second unless **DefaultStaggerRangeSecs=0.**

1.6.1.10 Agent.Log

Method	**Log**
Module	Agent
Library	Core
Action	Write a message to the Tachyon Agent log file.
Parameters	*Level* (string; optional, default *info*): The log level at which to write the message; one of: *trace*, *debug*, *info*, *warn* or *error*. These values are not case-sensitive. *Message* (string): The message to write to the log. Messages longer than 1024 characters will be truncated.

Method	**Log**
Return values	*Level* (string): The log level at which the message was written (as per *Level* parameter). *Message*(string): The message written to the log (as per the *Message* parameter).

Example	`Agent.Log(Level: "info", Message: "Starting file analysis");`
	
	`// etc.`
	
	`Agent.Log(Level: "info", Message: "Completed file analysis");`

Platforms	• Windows • Linux • MacOS
Notes	Useful for diagnosing/monitoring progress of an instruction. The Agent's log level setting determines whether the specified message actually appears in the log. For example, if the Agent's log level is configured as "info", then "debug" messages will not be visible in the log (but the output of the method will still return the Level and Message values).

1.6.1.11 Agent.GetKeyFiles

Module	Agent
Library	Core
Action	Returns names and paths of key files relating to the execution of the Tachyon Agent.
Parameters	(None)
Return values	*Configuration* (string): The full path of the file used to configure the Tachyon Agent. *Executable* (string): The full path of the executable file corresponding to the running Tachyon Agent.

Log (string): The full path of the Tachyon Agent log file.

DataRootDirectory (string): The directory used by the 1E Client for general data storage.

TemporaryDirectory (string): The directory actually used by the 1E Client for temporary files and directories.

Note

This is derived from either the configuration value if the specified setting is valid, or the default temporary directory.

The following table shows the OS-specific defaults used if the setting is omitted, blank, or invalid. The default is recommended.

Platform	Environment Variable(s)	Default
Windows	TMP TEMP	*%windir%\Temp* which is the default *%TEMP%* location for SYSTEM.
Linux	TMPDIR	/tmp

TemporaryDirectory is a configuration setting of the 1E Client.

Example

The following example reads the Tachyon Agent configuration file and shows the contained settings and values:

```
@keyFiles = Agent.GetKeyFiles
();
```

Module	Agent
	/* Bail if the Agent has no configuration file (see Notes below) */ SELECT * FROM @keyFiles WHERE Configuration != ""; EVALUATE; /* Read each line of configuration file */ @lines = FileSystem.GetFileByLine(FileName: @keyFiles.Configuration); /* Filter out comment entries, and parse remaining lines into Setting and Value */ SELECT LTRIM(REGEXPREPLACE("([^=]*)=(.*)", Content, "$1")) AS Setting , LTRIM(REGEXPREPLACE("([^=]*)=(.*)", Content, "$2")) AS Value FROM @lines WHERE LTRIM(Content) REGEXP "([^\\#])*=(.*)"; Uses ECMA262 format for regular expressions: http://www.cplusplus.com/reference/regex/ECMAScript/ http://en.cppreference.com/w/cpp/regex/ecmascript
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.12 ContentDistribution.Stage

Module	ContentDistribution
Library	Core
Action	Downloads content from a web server.
Parameters	*LszUrl* (string): The URL of a Nomad-style LSZ file that acts as a manifest for the content. *DirectoryUrl* (string): The URL of the top-level directory of the content. The directory can contain files and an arbitrary level of subdirectories. Only files listed in the LSZ manifest will be downloaded, but the LSZ file should have been generated to use the entire content. *ContentLocation* (string): Where to download the content to on the Tachyon agent host.
Return values	*Downloaded* (boolean): Whether download was successful or not. *NomadStatus* (string): One of • – "*Nomad not supported*" (for non-Windows) – "*Unknown*" (Windows, but did not get a chance to use Nomad) – "*Nomad successful*" (for download) – "*Nomad failed*" (Nomad not installed or suitable, or download with Nomad attempted but failed, but subsequent HTTP download may have succeeded) .
Example	`ContentDistribution.Stage(LszUrl:"http://fileserver.domain.local/Content/Tachyon_1d82db90-39ba-47c3-98b1-4483d20ac8f7_1.LsZ", DirectoryUrl:"http://fileserver.domain.local/Content/Tachyon_1d82db90-39ba-47c3-98b1-4483d20ac8f7", ContentLocation:"c:\hiddenshare\prestagedContent\patch12");`
Platforms	• Windows (also supports Nomad) • Linux • MacOS

Module	ContentDistribution
Notes	If *NomadContentDownloadEnabled* is set to *true* in the Agent's configuration file and Nomad is available and running then Nomad will be used for download. If Nomad is not available or fails then the Agent falls back to its internal HTTP provider.

1.6.1.13 ContentDistribution.StageAndExecute

Method	**StageAndExecute**
Module	ContentDistribution
Library	Core
Action	Similar to *Stage* (including optional Nomad integration), downloading content from a web server, but also executes a command on successful download.
Parameters	*LszUrl* (string): The URL of a Nomad-style LSZ file that acts as a manifest for the content. *DirectoryUrl* (string): The URL of the top-level directory of the content. The directory can contain files and an arbitrary level of subdirectories. Only files listed in the LSZ manifest will be downloaded, but the LSZ file should have been generated to use the entire content. *CommandLine* (string): A command to run if download is successful.
Return values	*Downloaded* (boolean): Whether download was successful or not. *NomadStatus* (string): One of • – "*Nomad not supported*" (for non-Windows) – "*Unknown*" (Windows, but did not get a chance to use Nomad) – "*Nomad successful*" (for download) – "*Nomad failed*" (Nomad not installed or suitable, or download with Nomad attempted but failed, but subsequent HTTP download may have succeeded) *ExitCode* (int): The exit code from the command (or the attempt to run the command if the command is not defined).

Method	**StageAndExecute**

Warning

Many Windows programs set a zero exit code even on failure.

Output (string): The output that the command sends to standard output and standard error, combined.

Example

```
 ContentDistribution.Stage(LszUrl:"http://fileserver.domain.local/
Content/Tachyon_1d82db90-39ba-47c3-98b1-4483d20ac8f7_1.LsZ",
                           DirectoryUrl:"http://fileserver.domain.
local/Content/Tachyon_1d82db90-39ba-47c3-98b1-4483d20ac8f7",
                           CommandLine:"msiexec /i installer.msi /
qn /l*v c:\hiddenshare\logs\program12\installation.log");
```

Platforms

- Windows (also supports Nomad)
- Linux
- MacOS

Notes

This action is typically used to download and then install software. The downloaded content is stored in a temporary location which is deleted on completion.

If you just want to execute a script or an install command for pre-existing content then use Scripting.Run or NativeServices.RunCommand methods instead.

Just before the CommandLine is executed, the working directory is (temporarily) set to where the content was downloaded to (which is a temporary directory). So, for example, if the downloaded content includes an installer file named "*app.msi*", then the CommandLine "*msiexec /i app.msi ...*" will work, i.e. you do not have to worry about deriving a full path.

Warning: Space for temporary files

Because *StageAndExecute* downloads content to a temporary area, there can be a problem with large content (e.g. GB in size) on non-Windows platforms if the default temporary directory, usually */tmp*, is in a filesystem that is not big enough. For example, on a Fedora system with 80 GB of disc, a default allocation for the *tmpfs* filesystem which houses */tmp* is only 500 MB. If this problem occurs, you will see an error like this in the Tachyon agent log:

```
INFO - Downloading "<remote_URL>" to file "/tmp/tachyon<aaaaaa>/<l
ocal_file>"
...
```

Method	**StageAndExecute**

```
ERROR - Error while trying to write to "/tmp/tachyon<aaaaaa>/<loca
l_file>": (28) No space left on device
```

The selection order for temporary file and directory locations on non-Windows platforms is as follows:

1. Environment variable *TMPDIR*, if set
2. The standard Linux/Unix location defined by *P_tmpdir*, usually, */tmp*
3. */tmp*
4. */var/tmp*

The Tachyon agent selects the first of these that is an existing directory.

Hence, if the error occurs (typically with a */tmp* that is too small), environment variable *TMPDIR* can be set to */var/tmp*, for example, which is usually on a filesystem with much more space assigned to it. However, since the agent runs as root daemon/service, it is not recommended to make any changes that would affect all root actions. Instead, change the way the daemon runs as follows. This describes Linux systems that run daemons using the systemd subsystem. As usual, use *sudo* commands to perform these actions.

First, after installation of the Tachyon Agent, edit */usr/lib/systemd/system/tachyon.agent.service* to add this line in the *[Service]* section:

```
Environment="TMPDIR=/var/tmp"
```

(or specify some other suitable directory for large temporary files).

Then run these commands to pick up the change and restart the Tachyon agent:

```
systemctl daemon-reload
systemctl restart tachyon.agent.service
```

1.6.1.14 Device.GetCriticality

Method	**GetCriticality**
Module	Device
Library	Core
Action	Gets the user-defined criticality level for this device.

Method	**GetCriticality**
Parameters	(None)
Return values	*Value* (integer): The (numeric) criticality value for this device (zero if undefined).

Example	`@c = Device.GetCriticality();`
	
	`@c = SELECT * FROM @c WHERE Value < 5;`
	
	`IF (@c)`
	`  // Perform action on devices with a criticality`
	`  // rating lower than five.`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	Use *Device.SetCriticality* to set the criticality value.

1.6.1.15 Device.GetInboundMappedDriveHistory

Method	**GetInboundMappedDriveHistory**
Module	Device
Library	Core
Action	Retrieves information about any connected shared folders or drives this device uses on remote servers.
Parameters	None
Return values	One row per instance of remote mapped drive usage at the time the share was established. A given server/share pair may appear in more than one row for different IP addresses or ports. • LastEstablished (datetime): The time at which the share usage was last established by this device. • *Port* (int): The port associated with the communication between client and server.

Method	**GetInboundMappedDriveHistory**
	• *ServerFqdn* (string): The server that the shared drive/folder was located on. • *ServerIpAddress* (string): The IP address of the server. • *ShareName* (string): The name of the shared drive on the server.
Example	`Device.GetInboundMappedDriveHistory();`
Platforms	• Windows
Notes	"Inbound" refers to the perspective of the device on which the Tachyon Agent is running, i.e. the shares in question are those provided by external servers to the device, *not* shares exported by the device.

1.6.1.16 Device.GetDisks

Method	GetDisks
Module	Device
Library	Core
Action	Retrieves information about the hardware disk drives
Parameters	(None)
Return Values	*DiskId* (string): The identifier of the disk

DiskName (string): The vendor-defined name of this disk

DiskType (string): The media type of the disk (e.g. *SSD*, *HDD*, *SCM* or *Unspecified* if not known)

DiskSerialNumber (string): The serial number of the disk

DiskSizeGb (string): The size of the disk in GB

DiskStatus (string): The status of the disk.

For Windows this is one of the *Status* values of the *Win32_DiskDrive* WMI class (*OK*, *Error*, *Degraded*, *Unknown*, *Pred Fail*, *Starting*, *Stopping*, *Service*, *Stressed*, *NonRecover*, *No Contact*, or *Lost Comm*), or *Not Available* if it cannot be found.

Example	`Device.GetDisks();`
Platforms	• Windows, but only from Windows 8 onwards • Linux • MacOS
Notes	Available from v8.0

1.6.1.17 Device.SetLocation

Method	**SetLocation**
Module	Device
Library	Core
Action	Sets the user-defined location for this device.
Parameters	*Location*(string): The location value to assign to this device (up to 127 characters). **Note** An empty value is okay, but the parameter itself is mandatory.
Return values	*Location*(string): The updated location value for this device.
Example	`Device.SetLocation(Location: "New York");`
Platforms	• Windows • Linux

Method	**SetLocation**
	• MacOS
Notes	Use *Device.GetLocation* to retrieve the assigned location value. Changes to location are automatically sent back to the Tachyon Server so that the assigned value can be used as part of a coverage expression.

1.6.1.18 Device.GetOutboundPrinters

Method	**GetOutboundPrinters**
Module	Device
Library	Core
Action	Retrieves information about any printers attached physically that might be shared
Parameters	None
Return values	A table containing a single row per attached printer • – *Attributes* (string): Some operating system details about the printer • "queued", "direct", "default", "shared", "hidden", "enable devq", "keep printed jobs", "do complete first", "work offline", "enable bidi", "raw only", "published", "fax" – *Location* (string): Whether the printer is "*local*" or "*remote*" – *Port* (string): The port associated with the printer. If *local* then this will be the physical port on this device, if *remote* then it is the IP Address of the server. – *PrinterName* (string): The name of the printer unique to the device – *ShareName* (string): The name under which the printer is shared from this device
Example	`Device.GetOutboundPrinters();`

Method	**GetOutboundPrinters**
Platforms	• Windows
Notes	

1.6.1.19 Device.GetSummary

Method	**GetSummary**
Module	Device
Library	Core
Action	Retrieves a summary of device information, corresponding to (a subset of) the data that is exchanged as part of the Agent's "online" message.
Parameters	(None)
Return values	*BiosGuid* (string): The Systems Management (SM) BIOS GUID of the device. *BiosVersion* (string): The version of the BIOS installed on the device *BootTime* (datetime): The last time the device was booted *ChassisType* (string): The device's physical chassis type (e.g. "Notebook") *CpuArchitecture* (string): The architecture of the device's (primary) CPU (e.g. "x64") *CpuType* (string): The device's (primary) CPU (e.g. "Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz") *DeviceType* (string): One of: "*Server*", "*Desktop*", "*Laptop*" or "*Mobile*", based on the observed hardware and operating system. Can be "*Unknown*" if this value cannot be derived. *Domain* (string): The Windows NetBIOS domain name, provided the device is domain-joined *Fqdn* (string): The fully-qualified DNS hostname of the device. *Manufacturer* (string): The device's chassis manufacturer *Model* (string): The device's chassis model

Method	**GetSummary**

NetBiosName (string): On Windows devices this is the NetBIOS computer name. On non-windows devices this column is null.

OperatingSystemGuid (string): The operating system GUID (e.g. "4C4C4544-0043-3210-804D-C3C04F4A4D32")

OperatingSystemName (string): The operating system distribution (e.g. Windows, Fedora, Ubuntu) that the device is running

OperatingSystemType (string): The family of operating system that the device is running; one of "Windows", "Linux" or "MacOS".

OperatingSystemVersionNumber (string): The dotted-numeric representation of the operating system version that the device is running (e.g. 7.1.0.500)

OperatingSystemVersionText (string): A string description of the operating system version (e.g. "Microsoft Windows Server 2008 R2 Enterprise")

OsArchitecture (string): The processor architecture of the operating system. The values depend on the *OperatingSystemType* as follows:

-
  - Linux: "*x86*" (32-bit) or "*x86_64*"; same as the result of the "*uname -m*" command.
  - MacOS: "*arm64*", "*x86*" (32-bit) or "*x64*".
  - Windows: "*x86*" (32-bit) or "*x64*".

RamMB (int): The amount of installed RAM in megabytes

SerialNumber (string): The device's serial number. (This is an additional column from version 24.1 onwards.)

TachyonGuid (string): The unique Tachyon identity (represented as a GUID) of this device

TimeZone (int): The device's current timezone offset from UTC in minutes (positive or negative).

VirtualizationPlatform (string): The name of the virtualization technology, if any, hosting the device's operating system

Example

```
Device.GetSummary();
```

Platforms

- Windows

Method	**GetSummary**
	• Linux • MacOS
Notes	See also Agent.GetSummary

1.6.1.20 Device.GetLocation

Method	**GetLocation**
Module	Device
Library	Core
Action	Gets the user-defined location for this device.
Parameters	(None)
Return values	*Location* (string): The location value for this device.

Example	`@loc = Device.GetLocation();`
	`@loc = SELECT * FROM @loc WHERE Location = "New York";`
	`IF (@loc)`
	`    // Perform action on devices in NY`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	Use *Device.SetLocation* to set the location value.

1.6.1.21 Device.SetCriticality

Method	**SetCriticality**
Module	Device
Library	Core
Action	Sets the user-defined criticality level for this device.

Method	**SetCriticality**
Parameters	*Value* (int): The (numeric) criticality value to assign to this device (between 0 and 255 inclusive).
Return values	*Value* (int): The updated criticality value for this device.
Example	`Device.SetCriticality(Value: 3);`
Platforms	• Windows • Linux • MacOS
Notes	Use *Device.GetCriticality* to retrieve the assigned criticality value. Changes to criticality are automatically sent back to the Tachyon Server so that the assigned value can be used as part of a coverage expression. The 1E platform and applications interpret these numeric values based on the *CriticalityMapping* global setting defined in the Tachyon Server. This setting provides a translation between the numeric value and the corresponding description.

1.6.1.22 Device.GetOutboundMappedDriveHistory

Method	**GetOutboundMappedDriveHistory**
Library	Core
Action	Retrieves information about any shared folders this device is exposing.
Parameters	None
Return values	A table containing one row per instance of local shared drive or folder usage. A server share may appear in more than one row for different client IP addresses or FQDN. • *ClientFqdn* (string): The FQDN of the client that was using the shared drive/folder • *ClientIpAddress* (string): The IP Address of the client that was using the shared drive/folder

Method	**GetOutboundMappedDriveHistory**
	• *FilePath* (string): The local path for current shares or the file that was being accessed for historical shares • *LastSeen* (datetime): The time at which the share usage was last seen by this device – historical share usages have the time of the usage – current share usages have the instruction execution time • *ShareName* (string): The name of the shared drive/folder on this server
Example	`Device.GetOutboundMappedDriveHistory();`
Platforms	• Windows
Notes	"Outbound" refers to the perspective of the device on which the Tachyon Agent is running, i.e. the shares in question are those exported by the device, *not* those provided to it by external servers.

1.6.1.23 Device.GetInboundPrinters

Method	**GetInboundPrinters**
Module	Device
Library	Core
Action	Retrieves information about any connected shared printer servers
Parameters	None
Return values	A table containing a row per attached printer • – *Attributes* (string): Some operating system details about the printer • "queued", "direct", "default", "shared", "hidden", "enable devq", "keep printed jobs", "do complete first", "work offline", "enable bidi", "raw only", "published", "fax" – *Location* (string): Whether the printer is "*local*" or "*remote*"

Method	**GetInboundPrinters**
	– *Port* (string): The port associated with the printer. If *Local* then this will be the physical port on this device, if *remote* then it is the IP Address of the server. – *PrinterName* (string): The name of the printer unique to the device – *ServerName* (string): The server that the attached remote printer is shared from. – *ShareName* (string): The name under which the printer is shared from this device
Example	`Device.GetInboundPrinters();`
Platforms	• Windows
Notes	

1.6.1.24 Device.GetProcessors

Module	Device
Library	Core
Action	Retrieves information about the CPUs plugged in to the motherboard of the device.
Parameters	(None)
Return values	*Vendor* (string): The vendor of the processor *Name* (string): The vendor-defined name of this processor package *CoreCount* (int): The number of cores inside this package *SpeedMhz*(int): The clock speed of the processor in its current operating mode From v5.1 onwards these columns also appear : *LogicalCoreCount* (int) : The number of logical cores inside this package *Hyperthreading* (bool): Indicates whether the cores are hyperthreaded, true for yes, false for no.

Module	Device
Example	`Device.GetProcessors();`
Platforms	• Windows • Linux • MacOS
Notes	*Name* is the immutable string that the CPU reports to SMBIOS. This often contains the publicly marketed and sold average speed of the CPU, which usually differs to the actual speed of the processor instance. *SpeedMhz* on Windows can change over time for one particular CPU. Modern CPU's can adjust their clock in response to heat and power from moment to moment. This parameter cannot be relied upon for scoping future instruction executions unless some kind of range of values is in the scope to cope with natural speed variance.

1.6.1.25 Discovery.LookupNetBiosNames

Method	**LookupNetBiosNames**
Module	Discovery
Library	Discovery
Action	Performs a NetBios reverse lookup on a list of IP addresses
Parameters	IpAddresses (string): A list of IPv4 addresses in string notation separated by commas.
Return values	*IpAddress* (string): The IP address that was looked up. *HostName* (string): The resolved host name.
Example	Returns one row, as the IPv6 address is ignored. `Discovery.LookupNetBiosNames(IpAddresses:"1.2.3.4,::1");`
Platforms	• Windows

Method	**LookupNetBiosNames**
Notes	This method will only resolve IPv4 addresses. IPv6 addresses will be ignored, and a warning will be logged. **Warning** This method may request information over the wire. As such, it shouldn't be sent to a lot of machines. Staggering is also a recommended practice.

1.6.1.26 Discovery.SendArpRequest

Method	**SendArpRequest**
Module	Discovery
Library	Discovery
Action	Performs an ARP request to retrieve the physical address for the supplied IPv4 address.
Parameters	IpAddress (string): The IPv4 address that the ARP request is intended to retrieve a physical address for.
Return values	*`IpAddress`* (string): The destination IP for the ARP request. MacAddress (string): The physical address that corresponds to *`IpAddress`*.
Example	Returns a MacAddress and an IpAddress in a single row. `Discovery.SendArpRequest(IpAddress:"1.2.3.4");`
Platforms	• Windows
Notes	This method will only attempt to send ARP requests for IPv4 addresses. IPv6 addresses will be ignored, and a warning will be logged. **Warning** This method may request information over the wire. As such, it shouldn't be sent to a lot of machines. Staggering is also a recommended practice. **Warning**

Method	**SendArpRequest**
	This method will never return anything other than *SuccessNoContent* or *Success*. If the ARP request is unsuccessful, the reason will be logged at *warn* level and *SuccessNoContent* will be returned.

1.6.1.27 Discovery.ScanPortsOnDevices

Method	**ScanPortsOnDevices**
Module	Discovery
Library	Discovery
Action	Probe a set of ports on a set of devices.
Parameters	*IpRanges* (string): A comma-separated list of IP address ranges that should be scanned for devices with ports. • – This parameter should be of the form "<ipAddressLow>-<ipAddressHigh>[,<ipAddressLow>-<ipAddressHigh>...]", with no spaces. • e.g. "1.2.3.4-1.2.3.100,2.3.4.0-2.3.4.255". – Should not be empty *Ports* (string): A comma-separated list of individual port numbers that should have a scan attempted on all the devices found. • – This parameter should be of the form "<port>[,<port>...]" • e.g. "80,443,127,128,25"

- Should not be empty

TimeoutPerPortMilliseconds (integer, optional, default 500): The maximum time period, in milliseconds, to wait for a port to respond before giving up. This needs to be tailored to suit the individual network to avoid giving up too soon and not learning anything.

- *TimeoutPerPortMilliseconds* <= *TimeoutPerPortMilliseconds* <= *TimeoutPerPortMilliseconds*

ConcurrentIps (integer, optional, default 8): The maximum number of devices undergoing port scanning at once. Most of the time this method will be scanning "ConcurrentIps" devices until there are not that many to work on where it drops off as they finish.

- *ConcurrentIpsMin* <= *ConcurrentIps* <= *ConcurrentIpsMax*

WaitBetweenPortsMilliseconds (integer, optional, default 500): The minimum time period, in milliseconds, to leave between the last successful port scan (ie anything but a timeout) and starting the next port scan, otherwise there will have already been a delay of length *TimeoutPerPortMilliseconds*, which generally would mean there is either no device at this IP address or a clever firewall is ignoring it.

Method	**ScanPortsOnDevices**
• – *WaitBetweenPortsMilliseconds*Min <= *WaitBetweenPortsMilliseconds* <= *WaitBetweenPortsMilliseconds*Max	
Return values	*Ip* (string): The IP address of the device in the IP-range the agent scanned *Port* (integer): The port on the device that was scanned *Result* (integer): The outcome of the scan of the port: Identifier / Meaning 0 / timeout 1 / rejected 2 / connected *Response* (string): A blank field reserved for future use
Example	`Discovery.ScanPortsOnDevices(IpRanges: "192.168.100.1-192.168.100.255", Ports: "22,23,25,53,80,443");`
Platforms	• Windows • Linux • MacOS
Notes	The default limits are configurable. The following configuration settings should be set to numbers only: • *Module.Discovery.TimeoutPerPortMilliseconds*Min (default 500)

Method	**ScanPortsOnDevices**
	• *Module.Discovery.TimeoutPerPortMilliseconds Max* (default 10,000) • *Module.Discovery.ConcurrentIpsMin* (default 1) • *Module.Discovery.ConcurrentIpsMax* (default 32) • *Module.Discovery.WaitBetweenPortsMilliseconds Min* (default 500) • *Module.Discovery.WaitBetweenPortsMilliseconds Max* (default 10,000) The following setting configures the maximum number of IP Addresses contained within a range; e.g. 10.10.10.1-10.10.10.254 would be a range of 254 IP addresses, so if *IpAddressRangeSizeMax* is set to 128 then it would reject 254 addresses, but would allow 10.10.10.2-10.10.10.129 • *Module.Discovery.IpAddressRangeSizeMax* (default 32) There is no minimum value to configure for this - a range could have zero IP addresses if it is specified.

1.6.1.28 FileContent.FindAndDelete

Method	**FindAndDelete**
Module	FileContent
Library	FileContent
Action	Deletes a string from a file.
Parameters	*FilePath* (string; required): The full path of the file. Environment variables in the form %variable% for windows, and $variable for non-windows operating systems will be expanded. *FindString* (string; required): The search string.

Method	**FindAndDelete**
	Backup (bool; optional, default *false*): Whether or not to backup the file prior to any deletions. A copy of the file will be made in the same directory with the *.bak* suffix. If this already exists, an incremental integer will be added to the filename after the extension. If a backup cannot be created, this method an error will be returned. *CaseSensitive* (bool, optional, default *true*): Whether or not to match case when searching for *FindString* in the file. *Occurrence* (string; optional, default "*All*"): How the deletion occurs. Valid values are "*First*", "*Last*" and "*All*". "*First*" will only delete the first occurrence matched, "*Last*" will only delete the last, and "*All*" will delete all matches.
Return values	*FilePath* (string): The full path of the file, with any environment variables expanded. *LineNumber* (int): The line number on which a deletion occurred. Lines are split on newline characters. *LineContent*(string): The content of the line after a deletion occurred. *BackupFileName* (string): The full path of the backup file created if *Backup* is true. A *NULL* is returned if *Backup* is false.

Example	Delete all occurrences of 'covers of queen songs are great!' regardless of case, making a backup of the file before the replacement occurs.
	
	`@FilePath = "C:\\temp\\Examples\\Example.conf";`
	`@stringToBeRemoved = "AuthenticationPolicy=optional";`
	`FileContent.FindAndDelete(FilePath: @FilePath, FindString: @string ToBeRemoved, Backup:true, CaseSensitive:false, Occurrence:"All");`

Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.29 FileContent.GetTorrentDescribedFiles (deprecated)

Method	**GetTorrentDescribedFiles (deprecated; will no longer be available after v8.1)**
Module	FileContent
Library	FileContent
Action	Returns all files described within the *info* and *files* structures contained inside a torrent file.
Parameters	*FilePath* (string; required): The full path of the torrent file.
Return values	*DescribedFileName* (string): The filename of a file described within the torrent file. *DescribedFileSizeBytes* (int): The size in bytes of a file described within the torrent file.
Example	Get the files described in a torrent called myTorrent.torrent. `FileContent.GetTorrentDescribedFiles(FilePath:"C:\\torrents\\myTorrent.torrent");`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.30 FileContent.WriteText

Method	**WriteText**
Module	FileContent
Library	FileContent
Action	Writes text to a file, overwriting any existing contents.
Parameters	*FilePath* (string): The full path of the file. If the file already exists its contents will be overwritten. Environment variables in the form %variable% for windows, and $variable for non-windows operating systems will be expanded.

Method	**WriteText**
	Text (string): The text string to be written. *Backup* (bool; optional, default *false*): Whether or not to backup the file prior to writing. If the file already exists, a copy of the file will be made in the same directory with the *.bak* suffix. If this backup file already exists, an incrementing number will be added after the extension. If a backup cannot be created, this method will return an error. *Encoding* (string, optional, default "*UTF8NOBOM*"): File content encoding. Valid values are: • "*ANSI*" • "*UTF8*" – UTF8 with a byte order mark. • "*UTF8NOBOM*" – UTF8 without a byte order mark. • "*UTF16LE*" – UTF16 little endian byte order. • "*UTF16BE*" -UTF16 big endian byte order.
Return values	*ExpandedFilePath* (string): The file path of the written file, with any environment variables expanded. *Success* (bool): Whether the method succeeded. *BackupFileName* (string): The full path of the backup file created if *Backup* is true. A *NULL* is returned if *Backup* is false.

Example	Create a file queen.txt, or backup and then overwrite an existing queen.txt with the supplied Text contents.
	
	`FileContent.WriteText(FilePath : "C:\\temp\\queen.txt", Text : "I want to break free!", Backup : true);`
	
	`The file contents become:`
	
	`I want to break free!`

Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.31 FileContent.FindAndReplace

Method	**FindAndReplace**
Module	FileContent
Library	FileContent
Action	Replaces a string in a file with another string.
Parameters	*FilePath* (string): The full path of the file. Environment variables in the form %variable% for windows, and $variable for non-windows operating systems will be expanded. *FindString* (string): The string to search the file for. *ReplaceString* (string): The string that will replace *FindString* *Backup* (bool; optional, default *false*): Whether or not to backup the file prior to replacing. A copy of the file will be made in the same directory with the *.bak* suffix. If this already exists, an incrementing int will be added after the extension. If a backup cannot be created, this method will return an error. *CaseSensitive* (bool, optional, default *true*): Whether or not to match case when searching for *FindString* in the file. *Occurrence* (string; optional, default "*All*"): How the replacement occurs. Valid values are "*First*", "*Last*" and "*All*". "*First*" will only replace the first occurrence matched, "*Last*" will only replace the last, and "*All*" will replace all matches.
Return values	*FilePath* (string): The full path of the file, with any environment variables expanded. *LineNumber* (int): The line number on which a replacement occurred. Lines are split on newline characters. LineContent (string): The content of the line after a replacement occurred. BackupFileName (string): The full path of the backup file created if *Backup* is true. A *NULL* is returned if *Backup* is false.
Example	Replace the first occurrence of 'AuthenticationPolicy=optional' with 'AuthenticationPolicy=required' regardless of case, making a backup of the file before the replacement occurs.

Method	**FindAndReplace**

	`@FilePath = "C:\\temp\\Examples\\Example.conf";`
	`@findString = "AuthenticationPolicy=optional";`
	`@replaceString = "AuthenticationPolicy=required";`
	`FileContent.FindAndReplace(FilePath: @FilePath, FindString: @findString, ReplaceString: @replaceString, Backup:true, CaseSensitive:false);`

Platforms	• Windows • Linux • MacOS
Notes	When a huge file is used with a lot of replacement, the Agent will consume high CPU and will take more time to process the instruction.

1.6.1.32 FileContent.FindAndAppend

Method	**FindAndAppend**
Module	FileContent
Library	FileContent
Action	Appends a string to the end of a file if it does ***not*** contain a specified string.
Parameters	*`FilePath`* (string): The full path of the file. Environment variables in the form *`%variable%`* for Windows and *`$variable`* for non-Windows operating systems will be expanded. *`FindString`* (string): The string to search the file for. *`AppendString`* (string): The string that will be appended to the end of the file. This can be the same as *`FindString`*. *`Backup`* (bool; optional, default *`false`*): Whether or not to backup the file prior to any appending. A copy of the file will be made in the same directory with the *.bak* suffix. If this already exists, an incrementing int will be added after the extension. If a backup cannot be created, this method will return an error. *`CaseSensitive`* (bool, optional, default *`true`*): Whether or not to match case when searching for *`FindString`* in the file.

Method	**FindAndAppend**
Return values	Returns 'Success - No Content' (empty table) if the string is found (file not updated), or the file is not found. If the file is found and updated the following are returned as a table: *FilePath* (string): The full path of the file, with any environment variables expanded. *LineNumber* (int): The line number on which the append occurred. Lines are split on newline characters. *LineContent* (string): The content of the line after appending *AppendString*. *BackupFileName* (string): The full path of the backup file created if *Backup* is true. A *NULL* is returned if *Backup* is false.

Example	Append the line 'AuthenticationPolicy=required' if it isn't present in example.conf, regardless of case, making a backup of the file before the append occurs.
	
	`@FilePath = "C:\\temp\\Examples\\Example.conf";`
	`@findString = "AuthenticationPolicy=required";`
	`@appendString = "AuthenticationPolicy=required";`
	`FileContent.FindAndAppend(FilePath: @FilePath, FindString: @findString, AppendString: @appendString, Backup:true, CaseSensitive:false);`

Platforms	• Windows • Linux • MacOS
Notes	Text is appended only if the find string does **not** exist in the file; this method should have been called FindElseAppend.

1.6.1.33 FileContent.AppendText

Method	**AppendText**
Module	FileContent
Library	FileContent
Action	Append text to a file.

Method	**AppendText**
Parameters	*FilePath* (string): The full path of the file. The file will be created if it doesn't exist. Environment variables in the form %variable% for windows, and $variable for non-windows operating systems will be expanded. *Text* (string): The text string to be appended to an existing file or written to a new file. *Backup* (bool; optional, default *false*): Whether or not to backup the file prior to writing. If the file already exists, a copy of the file will be made in the same directory with the *.bak* suffix. If this backup file already exists, an incrementing number will be added after the extension. If a backup cannot be created, this method will return an error. *NewLine* (bool; optional, default *false*) If true an O/S-specific line separator sequence will be written before the Text is written or appended. *Encoding* (string, optional, default "*UTF8NOBOM*"): File content encoding. Valid values are: • "ANSI" • "UTF8" - UTF8 with a byte order mark. • "UTF8NOBOM" - UTF8 without a byte order mark. • "UTF16LE" - UTF16 little endian byte order. • "UTF16BE" - UTF16 big endian byte order. If the file already exists and has content then an encoding should not be specified.
Return values	*ExpandedFilePath* (string): The full path of the written file, with any environment variables expanded. *Success* (bool): Whether the method succeeded. *BackupFileName* (string): The full path of the backup file created if *Backup* is true. A *NULL* is returned if *Backup* is false.

Example	Given an existing file *accounts.txt*, when the following code is executed:
	
	`@lines =`
	`  SELECT " { \"ac\": \"62781973\", \"sc\": \"05-21-07\", \"dob\": \"17-10-1993\" }" AS line`
	`  UNION SELECT ",{ \"ac\": \"01113201\", \"sc\": \"28-33-31\", \"dob\": \"12-01-1947\" }" AS line`

Method	**AppendText**

```
  UNION SELECT ",{ \"ac\": \"87290123\", \"sc\": \"20-35-27\", \"d
ob\": \"09-12-1996\" }" AS line
  UNION SELECT ",{ \"ac\": \"72098345\", \"sc\": \"21-09-37\", \"d
ob\": \"07-03-1993\" }" AS line;

FileContent.WriteText(FilePath : "C:\\TIMS\\Data\\accounts.txt", T
ext : "{ \"closed\" : [", NewLine : true, Backup: true);

FOREACH @line IN @lines DO
    FileContent.AppendText(FilePath : "C:\\TIMS\\Data\\accounts.tx
t", Text : @line, NewLine : true);
DONE;

FileContent.AppendText(FilePath : "C:\\TIMS\\Data\\accounts.txt",
Text : "]}", NewLine : true);
```

A backup *accounts.txt.bak* file is created containing the original file contents, and *accounts.txt* will now contain:

```
{ "closed" : [
 { "ac": "62781973", "sc": "05-21-07", "dob": "17-10-1993" }
,{ "ac": "01113201", "sc": "28-33-31", "dob": "12-01-1947" }
,{ "ac": "72098345", "sc": "21-09-37", "dob": "07-03-1993" }
,{ "ac": "87290123", "sc": "20-35-27", "dob": "09-12-1996" }
]}
```

Platforms

- Windows
- Linux
- MacOS

Notes

1.6.1.34 FileContent.GetTorrentInfo (deprecated)

Method	**GetTorrentInfo (deprecated; will no longer be available after v8.1)**
Module	FileContent
Library	FileContent
Action	Returns information contained inside a torrent file.
Parameters	*FilePath* (string; required): The full path of the torrent file.

Method	**GetTorrentInfo (deprecated; will no longer be available after v8.1)**
Return values	*Comment* (string): The comment field contained in the torrent, if it exists. *CreatedBy*(string): Typically describes the software used to create the torrent, if it exists. *CreationDate*(string): The creation date of the torrent. This is not the creation date on disk. *NumberOfSignatures*(int): The number of signatures contained within the *signatures* structure inside the torrent.
Example	Get the files described in a torrent called myTorrent.torrent. `FileContent.GetTorrentInfo(FilePath:"c:\\torrents\\myTorrent.torrent");`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.35 FileContent.GetTorrentAnnounceUris (deprecated)

Method	**GetTorrentAnnounceUris (deprecated; will no longer be available after v8.1)**
Module	FileContent
Library	FileContent
Action	Returns all announce URIs contained within the *announce* and *announce-list* structures contained inside a torrent file.
Parameters	*FilePath* (string): The full path of the torrent file.
Return values	*AnnounceUri* (string): An announce URI contained in the torrent file.
Example	Get on or more announce URI of a torrent called myTorrent.torrent.

Method	**GetTorrentAnnounceUris (deprecated; will no longer be available after v8.1)**
	`FileContent.GetTorrentAnnounceUris(FilePath:"c:\\torrents\\myTorrent.torrent");`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.36 FileSystem.CreateTempFile

Method	**CreateTempFile**
Module	FileSystem
Library	Core
Action	Creates an empty file in the OS temporary files location, reserving a unique name, and returns its path. In Windows system context the location is %TEMP% which resolves to **C:\Windows\temp** by default, and when running TIMS in user context (default) then %TEMP% resolves to **C:\Users\ <user> \AppData\Local\Temp**. In macOS and *nix OS the location is **/tmp**.
Parameters	*(None)*
Return values	*FilePath* (string): The full path of the file created.

Example	This writes some text to a temporary file and reads it, deleting the file when it is no longer needed.
	
	`@path = FileSystem.CreateTempFile();`
	`FileContent.WriteText(FilePath : @path, Text : "Hello");`
	`@text = FileSystem.GetFile(FilePath : @path);`
	`FileSystem.DeleteFileByName(FilePath : @path);`
	`SELECT Content FROM @text;`

Platforms	• Windows

Method	**CreateTempFile**
	• Linux • MacOS
Notes	The created file is not automatically deleted. As the example shows, an instruction using this method is responsible for deleting the file when it is no longer needed. The method can be called multiple times in an instruction and each time a file with a unique path is created. However, once the file has been deleted its name is available for re-use. The O/S temporary files location on Windows should not be confused with the 1E Client TemporaryDirectory setting (available in v5.1 onwards) which is %windir%\Temp by default.

1.6.1.37 FileSystem.DeleteFileByName

Method	**DeleteFileByName**
Module	FileSystem
Library	Core
Action	Deletes a file.
Parameters	*FilePath* (string): The path of the file to be deleted.
Return values	*Status* (string). Either: • "Deleted *FilePath*" • Last error string and execution error status if the file failed to be deleted. *Deleted* (boolean): *true* if the file existed and was successfully deleted, or *false* for any other reason (e.g. file could not be deleted due to permissions, file did not exist, etc.).
Example	Delete a Nomad lsz file. `FileSystem.DeleteFileByName(FilePath:"C:\\ProgramData\\1E\\NomadBranch\\1EP000AA_1.lsz");`

Method	**DeleteFileByName**
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.38 FileSystem.GetFilePermissions

Method	**GetFilePermissions**
Module	FileSystem
Library	Core
Action	Returns the set of permissions for a specified file or directory.
Parameters	*FilePath* (string) : The full path of the file or directory.
Return values	Each permission is returned as a row of these values: *UserOrRole* (string): A user-friendly representation of a username or role. So for example username *BUILTIN\ADMINISTRATORS* is the user-friendly representation of Security Id S-1-5-32-544. A Linux role is of the form *Owner*, *Group* or *Others*. *Inheritance* (string): The inheritance properties for this permission, for example *Inherited*, *None* or "*ForInheritanceOnly, FoldersInherit, FilesInherit*". Currently populated for Windows and macOS. *AccessClass* (string): An access class for this permission, typically *Allow*, *Deny*, *Audit* or *Alarm*. *AccessProperties* (string): The access properties for this permission, for example "*Read,Write,Execute*", *FullControl* or *Special*. The individual access property values are supported as follows:

Method **GetFilePermissions**

Access Property	Windows	Linux	Mac	Meaning if access allowed
AddFile			yes	(ACL) For a directory, add a file
AddSubDirectory			yes	(ACL) For a directory, add a subdirectory
Append			yes	(ACL) Append to a file
AppendData	yes			Append to a file
ChangePerms	yes			Modify the DACL in the object security descriptor for the file or directory
Chown			yes	(ACL) Change owner
CreateFiles	yes			For a directory, create files within
CreateFolders	yes			For a directory, create subdirectories
Delete	yes		yes	(ACL) For a directory, delete an existing file or subdirectory
DeleteChild	yes		yes	Windows: For a directory, delete it and all the files it contains (its children), even if the files are read-only. Non-Windows: (ACL) For a directory, delete a file or subdirectory.
Execute	yes	yes	yes	The file can be executed or the directory can be traversed and searched
FullControl	yes			Every right that Modify has, plus Change Permission and Take Ownership rights

Method	GetFilePermissions				
	List			yes	(ACL) The contents of the directory can be listed
	ListFolder	yes			For a directory, list the contents
	Modify	yes			Create, delete, change, and move files within their directory
	Read		yes	yes	The file/directory can be read
	ReadAttr	yes		yes	(ACL) Read basic (non-ACL) attributes
	ReadData	yes			The file can be read
	ReadExtAttr	yes		yes	(ACL) Read extended (i.e.named) attributes
	ReadPerms	yes			Read the security descriptor for the file or directory, excluding the information in the SACL
	ReadSecurity			yes	(ACL) Read ACL attributes
	SetGIDOnExe		yes	yes	If an executable file, run it with the group ID of the group
	SetUIDOnExe		yes	yes	If an executable file, run it with the user ID of the owner
	Special	yes			Some other access property
	Sticky		yes	yes	For a directory, only the owner and root can rename or delete objects within it
	Synchronize			yes	(ACL) Access the file locally at the server with synchronized reads and writes
	TakeOwnership	yes			Change the owner

Method	**GetFilePermissions**				
	TraverseFolder	yes			For a directory, traverse it
	Write		yes	yes	The file can be written to or the directory can have objects created within it
	WriteAttr	yes		yes	(ACL) Write basic (non-ACL) attributes
	WriteData	yes			The file can be written to
	WriteExtAttr	yes		yes	(ACL) Write extended (i.e. named) attributes
	WriteSecurity			yes	(ACL) Write ACL attributes

Example

```
FileSystem.GetFilePermissions(FilePath:"C:\\Test");
```

Platforms

- Windows
- Linux
- MacOS

Notes

Tachyon uses the same terms as Windows Explorer, such as *TakeOwnership* except that it presents these attributes as single words in a comma-separated list. In addition, Windows Explorer uses the word "*Special*" to report complex permissions such as readExtendedAttributes, readAttributes, readPermissions, etc. Tachyon accurately reports every attribute, because "*Special*" says only that standard permissions are not applied to the file.

Tachyon follows the Windows Explorer example and distinguishes between *Modify* and *FullControl*. *FullControl* containing every right that *Modify* does plus *ChangePerms* and *TakeOwnership* rights. With modify rights, a user can create, delete, change, and move files within their directories. But they cannot change the permissions or change the owner of these files. This ensures that permissions set on these files, by an administrator, will remain uniform.

GetFilePermissions has been restricted to only search the local computer for the translation of SIDs into "*domain**user*". If the Owner/Group SID identifies a user on a domain that is not the local machine, then the SID will be stringized into the "S-1-5-21-xxx-yyy" format. The reason for this local lookup only is that the Agents involved will all

Method	GetFilePermissions

be running the instruction at roughly the same time and thus this could be interpreted as a denial of service attack on the domain controllers.

For non-Windows endpoints, file mode permissions such as *drwsrwsrwt* are presented as minimal ACLs with the three roles *Owner*, *Group* and *Other* being presented. In addition, if any special modes are set for the file or directory then a Special role is also included which can include the set GID on execution, set UID on execution and sticky bit. For more information on these bits, a kapeli.com/cheat_sheets/Unix_File_Permissions.docset/Contents/Resources/Documents/indexrea snably good explaation is .

ACL support was added in the Linux 2.6 kernel and is available on a filesystem basis (ext2, ext3, ext, IBM JFS, XFS, reiserfs). Linux distributions typically offer ACL as an optional install feature, for example 'yum install acl' on Centos/Red Hat. Also, the file system will need to be mounted with the acl option. Linux ACL support is POSIX 1003.1e DS 17 compliant. This solution only offers read, write, execute permissions on a per user or per group basis, plus directory ACL attribute inheritance. The Linux solution is inferior to either MacOSX ACLs or Windows ACLs. Typically, ACLS are set using **setfacl** and displayed using **getfacl**.

ACL support is enabled by default for MacOSX Leopard onwards and the support is much richer than the Linux ACL solution, it is quite similar to Windows ACL solution. Typically, ACLs are set using **chmod** and displayed using **ls -le**. A good description of MacOSX ACL support is developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemDetails/FileSystemDetails.html#//apple_ref/doc/uid/TP40010672-CH8-SW20.

ACL support for Windows is rich and comparable to MacOSX ACL support. A good description is technet.microsoft.com/en-us/library/cc783530(v=ws.10).aspx.

A good table of operating system/file system support for ACLs is www.ibm.com/support/knowledgecenter/en/SSGSG7_7.1.1/com.ibm.itsm.client.doc/c_bac_aclsupt.html.

1.6.1.39 FileSystem.GetFileByLine

Method	GetFileByLine
Module	FileSystem
Library	Core

Method	**GetFileByLine**
Action	Returns the lines of the given text file(s) matching the supplied regular expression pattern.
Parameters	*FilePath* (string): The full path of the file. Similar to the GetFile method, the path can include environment variables represented by the local shell format (e.g. *%env%* on Windows), and wildcards such as '*' for multiple characters and '?' for a single character. *RegExp* (string; optional, default: empty string): A regular expression pattern in ECMA-262 ECMAScript format for the lines to match. If not supplied, every line will be returned.
Return values	A maximum of 50,000 lines is returned. For each line of each file matching the pattern found: • *FileName* (string): The full path, with any wildcards and environment variables expanded. • *LineNumber* (int): The line number, starting at 1. • *Content* (string): The content of the line that contains a match for the regular expression.
Example	Search the ConfigMgr WUAHandler log for any errors. `FileSystem.GetFileByLine(FileName:"c:\\Windows\\CCM\\Logs\\WUAHandler.log", RegExp:"[eE]rror");` By default, the search is case sensitive. Here we are using RexExp to search for both Error and error.
Platforms	• Windows • Linux • MacOS

Method	**GetFileByLine**
Notes	Do not use on a binary file - the returned *Content* will probably be corrupted. A good site to help understand basic regular expressions is https://regexr.com/ Uses ECMA262 format for regular expressions: http://www.cplusplus.com/reference/regex/ECMAScript/ http://en.cppreference.com/w/cpp/regex/ecmascript

1.6.1.40 FileSystem.GetDirectoryHash

Method	**GetDirectoryHash**
Module	FileSystem
Library	Core
Action	Returns the hash of a directory and its entire contents, including the contents of subdirectories recursively.
Parameters	*DirectoryPath* (string): The full path of the directory. *Algorithm* (string): Which algorithm to use to perform the hashing. Possible values: • *SHA1* • *SHA256* • *MD5*
Return values	*SizeBytes* (int): The size of the all files in the directory *Hash* (string): The SCCM 2012 hash of the directory, as there is no standard method of calculating directory hashes. On non-Windows machines this returns a hash that is different to those calculated on a Windows machine for the same data, but the result is consistent. This is typically intended to be used to check a directory's contents against another's.
Example	`FileSystem.GetDirectoryHash(DirectoryPath:"c:\\tmp", Algorithm:"MD5");`
Platforms	• Windows

Method	**GetDirectoryHash**
	• Linux • MacOS
Notes	Returns *SuccessNoContent* if a file is supplied as the *DirectoryPath* parameter. On Windows a remote directory is not supported, neither in UNC format nor using a mapped drive. This is not treated as an error but no content is returned.

1.6.1.41 FileSystem.GetDigitalSignature

Method	**GetDigitalSignature**
Library	Core
Action	Returns all certificates used in an authenticode signature that are not used for timestamping.
Parameters	*FilePath* (string): The full path of the file.
Return values	*CertificateIndex*(string): An index for each certificate chain returned. For example, if you had only one signature on the file, there may be multiple rows returned (as the certificate chain may be long), but all rows returned would have a *CertificateIndex* of 0. This can be used to isolate a particular certificate chain. This is a zero indexed number. *CertificateType* (string): The type of the certificate. Possible values: "Signing", "Intermediate", "Root" and "Self-signed". **Note** The CertificateType return value is inferred from the depth of the certificate in the chain built by the device. A depth 0 certificate can be marked as 'Self-signed' if no other certs in its trust chain can be found.

Depth (string): The depth of a certificate in a certificate chain. This starts from the certificate used to sign the file, which is 0. The next certificate in the chain is 1, and so on. I.e. a zero-indexed number. The Depth return value builds a certificate trust chain.

> **Note**
>
> If a certificate chain cannot be built on a device, for example if certificates are missing from the certificate store, the chain returned may be incorrect and will reflect this. This will also affect the *CertificateType* return value.

EffectiveDate (string): The date at which the certificate becomes valid ('*NotBefore*').

ExpiryDate (string): The date at which the certificate is no longer valid ('*NotAfter*').

FileName (string): The full path of the file.

HashAlgorithm (string): The algorithm of the hash used to create the digital signature. If the hashing algorithm used is SHA-1, SHA-256, SHA-384 or SHA-512, the return values will be "SHA1", "SHA256", "SHA384" and "SHA512" respectively. Other hashing algorithms will return an OID, such as "1.2.840.113549.1.1.9". These OIDs are searchable online, on sites such as oidref.com.

Issuer (string): The Issuer field of the certificate.

SerialNumber (string): The serial number of the certificate. This is supposed to be (according to RFC5280) a positive integer assigned by the issuing CA that is unique. It is a

Method	**GetDigitalSignature**
nice way to identify a certificate if you are dealing with a single CA. **Note** If you are dealing with multiple CAs, this is *not* a good way to specify a cert as this field can be zero. *SignatureStatus* (string): "*Signed*" if the certificate is signed otherwise "*Unsigned*". *Subject* (string): The Subject field of the certificate, containing the Common Name of the certificate. *Thumbprint* (string): The SHA1 hash of the certificate content and the certificate serial number.	
Example	`FileSystem.GetDigitalSignature(FilePath:"c:\\tmp\\SomeProgram.exe");`
Platforms	• Windows
Notes	Does *not* return the timestamping certificates.

1.6.1.42 FileSystem.GetFile

Method	**GetFile**
Module	FileSystem
Library	Core
Action	Returns the content of a text file.
Parameters	*FilePath* (string): The full path of the file. The path can include environment variables represented by the local shell format (e.g. *%env%* on Windows), and wildcards such as '*' for multiple characters and '?' for a single character.

Method	**GetFile**
Return values	If nothing is found then the method returns success but no rows, this is because a complex instruction script may want to continue to execute rather than fail abruptly. For each file found: • *FileName* (string): The full path, with any wildcards and environment variables expanded. • *Content* (string): The full content of the file.

Example	`FileSystem.GetFile(FilePath:"c:\\tmp\\i*.txt");`
	
	`FileSystem.GetFile(FilePath:"c:\\tmp\\ip?.txt");`
	
	`FileSystem.GetFile(FilePath:"%programdata%\\1E\\Client\\1E.Client.log");`

Platforms	• Windows • Linux • MacOS
Notes	Do not use on a binary file - the returned *Content* will probably be corrupted. This method accepts ANSI encodings for files. Results are unpredictable for other encodings. For example, UCS2-LE (UTF16-LE) files will only be partially returned, resulting in text such as '1' in the Tachyon Explorer. There is a limit of up to **10MB** on the size of the file to be fetched. If a file is this size or larger, then its contents will not be fetched. If no files are fetched due to this limit, the method fails. But it is not an error if no files match the *FilePath*, although no results are returned in that case.

1.6.1.43 FileSystem.GetFilesInFolder (deprecated)

Method	**GetFilesInFolder (deprecated from v3.2)**
Module	FileSystem
Library	Core
Action	Returns the filename, size, created time and modified time of each file in a directory.

Method	**GetFilesInFolder (deprecated from v3.2)**
Parameters	*Directory* (string): Full path of the directory whose contents will be returned. *Recursive* (boolean; optional, default *true*): Whether or not to return the contents of subdirectories.

Return values	If *Directory* does not exist then an ExecutionError status is returned along with the following string:
	
	`Directory '<directory>' does not exist`
	
	or
	
	`Folder '<folder>' does not exist`
	
	If *Directory* is actually a file then an ExecutionError status is returned along with the following string:
	
	`Directory '<directory>' is not a directory`
	
	or
	
	`Folder '<folder>' is not a folder`
	
	If *Directory* itself cannot be opened (exceptions will be logged by the agent) or no files are found then this is considered a successful execution but no results are returned.
	
	If *Directory* itself can be opened and Recursive is true but then a subdirectory or file cannot be accessed then those items will be skipped in the results (the agent will log warnings).
	
	Otherwise if any files are found then the result returned will have this form:
	
	• *FileName* (string): The full path of the file
	• *FileSize* (int): The size of the file in bytes
	• *DateCreated* (string): The creation date of the file
	• *DateModified* (string): The last modified date of the file

Example	`FileSystem.GetFilesInFolder(Directory:"c:\\tmp");`
	
	`FileSystem.GetFilesInFolder(Directory:"c:\\tmp", Recursive:false);`

Platforms	• Windows

Method	**GetFilesInFolder (deprecated from v3.2)**
	• Linux • MacOS
Notes	The DateCreated concept does not exist on some operating systems and consequently this field will be blank where it does not apply. Where the agent cannot access *Directory* itself then the instruction will return ExecutionError. The method has been deprecated from v3.2. Use GetFilesInDirectory instead.

1.6.1.44 FileSystem.GetHash (deprecated)

Method	**GetHash (deprecated)**
Module	FileSystem
Library	Core
Action	Returns the hash of a file or directory.
Parameters	*Path* (string; optional): The full path of the file or a directory. *Algorithm* (string): Which algorithm to use to perform the hashing. Possible values: • *SHA1* • *SHA256* • *MD5*
Return values	If a directory is named, the following are returned: • *DirectoryName* (string): The full path of the directory • *DirectorySizeBytes* (int): The total size of the all files in the directory • *DirectoryHash* (string): The SCCM2012 folder hash, as there is no standard method of calculating folder hashes. On non-Windows machines, this returns a hash that is different to those calculated on a Windows machine for the same data, but the result is consistent. This is typically intended to be used to check a folder's contents against another's.

Method	**GetHash (deprecated)**
	Otherwise, if a file is named, the following are returned: • *FileName* (string): The full path of the file • *FileSizeBytes* (int): The size of the file in bytes • *FileHash* (string): The hash of the file

Example	`FileSystem.GetHash(Path:"C:\\tmp\\ips.txt", Algorithm:"MD5");`
	`FileSystem.GetHash(Path:"C:\\tmp\\ips.txt", Algorithm:"SHA1");`
	`FileSystem.GetHash(Path:"C:\\tmp", Algorithm:"SHA256");`

Platforms	• Windows • Linux • MacOS
Notes	This method was deprecated in 3.1. Use GetFileHash and GetDirectoryHash instead.

1.6.1.45 FileSystem.GetFileHash

Method	**GetFileHash**
Module	FileSystem
Library	Core
Action	Returns the hash of a file.
Parameters	*FilePath* (string): The full path of the file. *Algorithm* (string): Which algorithm to use to perform the hashing. Possible values: • *SHA1* • *SHA256* • *MD5*
Return values	*SizeBytes* (int): The size of the file *Hash* (string): The hash of the file

Method	**GetFileHash**
Example	`FileSystem.GetFileHash(FilePath:"c:\\tmp\\ips.txt", Algorithm:"MD5");`
Platforms	• Windows • Linux • MacOS
Notes	Returns *`SuccessNoContent`* if a directory is supplied as the *`FilePath`* parameter.

1.6.1.46 FileSystem.FindFileBySizeAndHash

Method	**FindFileBySizeAndHash**
Module	FileSystem
Library	Core
Action	Finds a file by its size and hash.
Parameters	*`Size`* (int): the size in bytes of the file. *`Hash`* (string): the hash of the file. *`Fast`* (boolean; optional, default *`true`*): On Windows chooses between the fast search technology or the previous slower (but fully reliable) technology. Always used for non-Windows. *`Algorithm`* (string, optional, default "SHA256"): Selects the hashing algorithm used by this method. Supported values: • SHA1 • SHA256 • MD5 This must match the *`Hash`* parameter; if you specify an MD5 hash, this parameter must also be set to MD5. *`TimeoutSecs`* (integer; optional, default *`600`*): If the operation has not completed within the specified time period, it will fail with an error.

Method	**FindFileBySizeAndHash**

Exclude (Table, optional): A single column table where each row contains a root relative directory path that you want to exclude from recursive search.

> **Note**
>
> On Windows, excluded paths can contain drive letters on Windows (e.g. C:\\), to skip scanning whole volumes.
>
> Ensure the exclude paths are complete paths starting from root.

Return values

For each file found:

- *FileName* (string): The full path.

Example

This will perform a fast search, looking for a file with the specified SHA-256 hash and size:

```
FileSystem.FindFileBySizeAndHash(Size:402, Hash:"8eda0d394c2370c4a
c8562d5ae6b5e7b01f347940bc58a65ca873c33a33475a3");
```

This will perform a slow search, looking for a file with the specified MD5 hash and size:

```
FileSystem.FindFileBySizeAndHash(Size:999, Hash:"08ad2e5d7538091fb
bfd352bb323e238", Fast:false, Algorithm:"MD5");
```

```
@exclude =
SELECT
        column1 AS Path
FROM
        ( VALUES
                ("C:\\"),
                ("E:\\IgnoredFolder")
        );

FileSystem.FindFileBySizeAndHash(Size:402, Hash:"8eda0d394c2370c4a
c8562d5ae6b5e7b01f347940bc58a65ca873c33a33475a3",Exclude:@exclud
e);
```

Platforms

- Windows
- Linux
- MacOS

Notes

This will search all fixed disks which is an expensive process and may take some time.

Method	**FindFileBySizeAndHash**
	Currently files locked by some other process cannot be hashed and therefore cannot be reported as being found. This is not supported. This will not search for files on remote machines. This works slightly differently on Windows and Non-Windows platforms when the given file or directory where the file resides does not have any permissions assigned/available. Windows platforms do not report such locked-down files whereas non-Windows do report the files irrespective of which level permission has been denied (file or directory). This happens because the 1E.Client daemon runs as "root" (superuser).

1.6.1.47 FileSystem.GetFilesInDirectory

Method	**GetFilesInDirectory**
Module	FileSystem
Library	Core
Action	Returns the filename, size, created time and modified time of each file in a directory.
Parameters	*`Directory`* (string): Full path of the directory whose contents will be returned *`Recursive`* (boolean; optional, default *`true`*): Whether or not to return the contents of subdirectories. *`Limit`* (integer, optional, default *`8192`*): Return no more than this number of files; *`0`* means all files (i.e. no limit). *`Exclude`* (Table, optional): A single column table where each row contains a directory path relative to the 'Directory' parameter that you want to exclude from recursive search. **Note** On Windows a remote directory is not supported as a mapped drive, although UNC format does work. Ensure that the exclude paths are relative to the directory parameter.
Return values	If *`Directory`* does not exist (which can happen if the path format is not suitable for the device's operating system) or it is actually a file then an error status is returned with a suitable error message.

Method	**GetFilesInDirectory**
	If *Directory* itself cannot be opened (exceptions will be logged by the agent) or no files are found then this is considered a successful execution but no results are returned. If *Directory* itself can be opened and Recursive is true but then a subdirectory or file cannot be accessed then those items will be skipped in the results (and the Agent will log warnings). Otherwise if any files are found then the result returned for each will have this form: • *FileName* (string): The full path of the file • *FileSize* (int): The size of the file in bytes • *DateCreated* (string): The creation date of the file • *DateModified* (string): The last modified date of the file

Example	`FileSystem.GetFilesInDirectory(Directory:"c:\\tmp");`
	
	`FileSystem.GetFilesInDirectory(Directory:"c:\\tmp", Recursive:false);`
	
	`@exclude =`
	`SELECT`
	`column1 AS Path`
	`FROM`
	`( VALUES`
	`(".\\tmp1"),`
	`(".\\tmp2\\SubFolder")`
	`);`
	
	`FileSystem.GetFilesInDirectory(Directory:"C:\\tmp",Exclude:@exclude);`

Platforms	• Windows • Linux • MacOS
Notes	The DateCreated concept does not exist on some operating systems and consequently this field will be blank where it does not apply. Where the agent cannot access the directory itself then the instruction will return ExcutionError.

Method	**GetFilesInDirectory**

Important

OneDrive can report a file creation time after the last modified time.

1.6.1.48 FileSystem.GetVersionInfo

Method	**GetVersionInfo**
Module	FileSystem
Library	Core
Action	Returns version and some other details about a file.
Parameters	*FilePath* (string): The full path of the file. *Language* (string; optional): The language from which to retrieve strings, as a hexadecimal identifier; for example, for en-US, *Language* would be "*0409*". If this is supplied and the strings do not exist in the language, empty strings will be returned. If this is not supplied, the agent will attempt to retrieve strings in the first language and codepage specified in *VarFileInfo\Translation*. If this does not work, en-US (*0409*) will be tried.
Return values	*FileName* (string): The full path of the file *FileDescription* (string): The description of the file. *FileVersion* (string): The version of the file *ProductVersion* (string): The version of the product. File and Product versions may differ. *ProductName* (string): The product name. *OriginalFilename* (string): The file's original name.
Example	`FileSystem.GetVersionInfo(FilePath:"C:\\Program Files\\1E\\Tachyon \\Agent\\Tachyon.Agent.Language.dll");`
Platforms	• Windows
Notes	• File version infoproperties are based on version resource information built into the file. Version resources are often built into binary files such as .exe or .dll files, but text files do not have version resource information.

Method	**GetVersionInfo**
	• Version information is cached as an optimization from v4.2, but the cache is not retained when the 1E Client shuts down. A cache entry is updated if the modified time of the file has changed.

1.6.1.49 FileSystem.PathExists

Method	**PathExists**
Module	FileSystem
Library	Core
Action	Checks whether a path exists.
Parameters	*Path* (string): The path of the file system object. The path can include environment variables represented by the local shell format (e.g. %env% on Windows), and wildcards such as '*' for multiple characters and '?' for a single character.
Return values	If nothing is found then the method returns success but no rows; this is because a complex instruction script may want to continue to execute rather than fail abruptly. For each file or directory found: • *FullPath* (string): The full path, with any wildcards and environment variables expanded. • *IsDirectory* (bool): Represents if the path points to a file or directory.

Example	`FileSystem.PathExists(Path:"c:\\tmp\\i*.txt");`
	`FileSystem.PathExists(Path:"c:\\tmp");`
	`FileSystem.PathExists(Path:"%programfiles%\\1E\\Tachyon\\TIMS\\Tachyon.InstructionManagementStudio.exe");`

Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.50 FileSystem.FindDirectoryByName

Method	**FindDirectoryByName**
Module	FileSystem
Library	Core
Action	Finds directory paths given a leaf directory name.
Parameters	DirectoryName (string): The name of the directory to be found. This cannot contain path separators and therefore is always just the leaf node of the path (the deepest part of the path). **Note** The name can include wildcard characters ? and *. Case insensitive wildcards are supported only on Windows. *Fast* (boolean; optional, default *true*): Selects either the fast NTFS search technology or the previous,slower (but fully reliable) technology. **Note** Unix *Fast=false* and NTFS *fast=true* are similar order timescales (unless the Unix method comes across a mounted windows share); NTFS *Fast=false* is significantly slower. *TimeoutSecs* (integer; optional, default *600*): If the operation has not completed within the specified time period, it will fail with an error. *Exclude* (Table, optional): A single column table where each row contains a root relative directory path that you want to exclude from recursive search. **Note** On Windows, excluded paths can contain drive letters on Windows (e.g. C:\\), to skip scanning whole volumes. Ensure the exclude paths are complete paths starting from root.
Return values	For each directory found: • *Path* (string): The full path of the directory.
Example	`FileSystem.FindDirectoryByName(DirectoryName:"wibble", Fast:true);` The above method call might find the following on relevant platforms: *C:\Program Files\Acme Corp\Wibble*

Method	**FindDirectoryByName**

	/usr/local/wibble
	
	But it will never find these:
	
	C:\Program Files\Acme Corp\Wibble\bin
	
	/usr/local/wibble/bin
	
	`@exclude =`
	`SELECT`
	`column1 AS Path`
	`FROM`
	`( VALUES`
	`("C:\\"),`
	`("E:\\IgnoredFolder")`
	`);`
	
	`FileSystem.FindDirectoryByName(DirectoryName:"wibble",Exclude:@exclude);`

Platforms	• Windows • Linux • MacOS
Notes	This method searches all fixed disks, which is a resource expensive process and may take some time if *Fast=false*. Symbolic links are ignored on non-Windows OS.

1.6.1.51 FileSystem.FindFileByName

Method	**FindFileByName**
Module	FileSystem
Library	Core
Action	Finds a file by name.
Parameters	*FileName* (string): The name of the file to be found. The name can include wildcard characters ? and *. **Warning**

Method	**FindFileByName**
	Using a wildcard such as *FileSystem.FindFileByName(FileName:"*", Fast:true)* is never a good idea as this will cause all files to be returned and could have negative an effect on system performance. Wherever possible use specific wildcards to identify the files needed. **Note** Case insensitive wildcards are supported only on Windows.
Fast (boolean; optional, default *true*): chooses between the fast Windows NTFS search technology or the previous slower (but fully reliable) technology **Note** Ignored for non-NTFS filesystems. Unix *Fast=false* and NTFS *Fast=true* are similar order timescales (unless the Unix method comes across a mounted windows share), NTFS *Fast=false* is significantly slower.	
Recurse (boolean; optional, default *true*): Recurse into sub-folders. If this is set to false and no folder is specified, only the root folder of each drive/mount-point will be searched. If this is set to false and a folder is specified, only that folder on each drive/mount-point will	

be searched (if there is such a folder on that drive/mount-point).

> **Note**
>
> Turning off recursion will force use of safe (non-fast) search mode, even if fast mode is requested. However, since folders are then not recursed into, a non-recursive search ought in most cases to complete quite quickly.

Folder (string; optional, defaulting to the root folder): The folder to search in. The folder must be a 'root relative' path e.g. *\\windows\\system32* or */var/log*. The search will look in such folders on each drive (Windows) or mount-point (Unix). The folder path may include . and .. parts, which will be normalised into the equivalent canonical path e.g. *\\windows\\system\\..\\system32* will search in *\\windows\\system32*.

> **Note**
>
> Specifying a folder will force use of slow search mode,

Method	**FindFileByName**
even if fast mode is requested. *TimeoutSecs* (integer; optional, default *600*): If the operation has not completed within the specified time period, it will fail with an error. *Exclude* (Table, optional): A single column table where each row contains a root relative directory path that you want to exclude from recursive search. **Note** On Windows, excluded paths can contain drive letters on Windows (e.g. C:\\), to skip scanning whole volumes. Ensure the exclude paths are complete paths starting from root.	
Return values	For each file found: • *FileName* (string): The full path. • *Size* (int): The size in bytes. • *Hash* (string): The SHA-256 hash of the contents of the file. **Old unicode filenames and v5.1+** Disks that contain files that could have been created over two decades ago could well have filenames that were valid UCS-2 characters (in the range 0xd800 →

0xdfff), but that are not actually UTF-16 surrogate pairs, or just plain encoding may incorrectly exist on the file system quite happily, as NTFS allows them to still exist.

However, when these filenames are converted to UTF-8, which is the preferred internal representation in the 1E Client, then they fail to translate exactly, so it is possible to lose the original reference to the exact file name. This translation problem will cause a warning like this to be issued in the client logs:

```
2020-03-30 14:50:37,048 WARN - Invalid UTF-16 seque
nce found starting at offset 24 just after this UTF
-8 version [C:\dev\research\ucs2\bad??]
```

Because the logging is also in UTF-8 then the 1E Client cannot write the UTF-16 character into the log, so the post conversion UTF-8 characters are presented and as in this case, the incorrect unicode values have been replaced with question marks because they are not UTF-16 so are not possible to convert to UTF-8; this just indicates where the issue is in the string.

This is fundamental and applies to any string conversions in general, not just filenames specifically, but this method (and similar ones) is likely to find badly formed UTF-16 on the aged disk file system.

When this occurs in this method, then the result set will contain a row like this:

FileName	Size	Hash
c:\dodgy\unic ?de.t?t	429496 7295	cann ot-initia lise-hash ing

This is because the filename cannot actually be found once it has been mutated into UTF-8 with missing character markers.

The likelihood of encountering file names like this is very low, they have to be constructed by a scripting

Method	**FindFileByName**
	language, and are unlikely to work with most modern programs anyway. For further context, Windows started using UCS-2 from Windows NT 3.0 (1993) onwards and switched to UTF-16 from Windows 2000 (NT 5.0) but obviously any UCS-2 filenames had to remain usable. However this is ancient history.

Example	`FileSystem.FindFileByName(FileName:"ips.txt", Fast:true);`
	`@exclude = SELECT column1 AS Path FROM ( VALUES ("C:\\"), ("E:\\IgnoredFolder") );`
	`FileSystem.FindFileByName(FileName:"ips.txt", Exclude:@exclude);`

Platforms	• Windows • Linux • MacOS
Notes	This will search all fixed disks which is an expensive process and may take some time. Where possible, use a specific folder search and/or disable recursion to speed up the search. On Non-Windows, symbolically linked files will be ignored.

1.6.1.52 FileSystem.GetFileDetails

Method	**GetFileDetails**
Module	FileSystem
Library	Core
Action	Retrieves operating system attributes related to the specified file.

Method	**GetFileDetails**
Parameters	*FilePath* (string): The full path of the file. *ComputeHash* (boolean; optional, default *false*): Whether to calculate the Hash for the file. **Note** FilePath must not contain wildcards; they are assumed to be part of the FilePath.
Return values	If *FilePath* is an empty string then an *InvalidParameter* error is returned along with the following string: `FilePath parameter should not be empty` If *FilePath* does not exist or is invalid (both the same situation depending on the platform) then a Success(no content) status. If FilePath itself cannot be accessed (exceptions will appear in the agent log) or no files are found then this is considered a successful execution but no results are returned. Otherwise, a single row containing the following columns will be returned: • *FilePath* (string): The full path. • *FileSystemItem* (string): The type of item the path leads to: *file/directory/symbolic link/block device/character device/fifo/socket/unknown* • *Size* (int): The size in bytes if a file, blank if a directory. • *Hash* (string): If requested and this path is a file, then this column is the SHA-256 hash of the contents of the file. A symbolic link is a link to a file, not a file. • *Owner*(string): The account that is the owner of the file system item represented in the agent's operating system specific way. See **notes** below. • *Group* (string): The group account that is considered to own the file system item represented in the agent's operating system specific way. See **notes** below. • *DateCreated* (datetime): The creation date of the file system item in Windows in ISO 8601 UTC format, or blank if non-Windows.

- *DateModified* (datetime): The last modified date of the file system itemin ISO 8601 UTC format.

Attributes (string): A concatenated list of file attributes from the following, attributes mostly apply to files and directories, so 'file' can be substituted by 'directory' below.

The attributes available does depend on what format the file system is, ext2 and above support attributes most systems will default to ext4 if they have it, but obviously it depends on exactly what the fs type is of the volume that is mounted under the path specified. For MacOS file flags are displayed as Attributes, however extended attributes are not currently displayed.

Attribute	Windows	Linux	Mac	Meaning
read-only	yes			the file can only be read, this is independent of the security permissions
hidden	yes		yes	the file will not appear unless 'show hidden files' is enabled in explorer
system	yes			the file is considered to be a vital part of the operating system
archive	yes			the file has changed since the last system backup (which would clear this flag next time it is backed up)
device	yes			the file is actually a conduit for a device
temporary	yes			the file is marked as temporary, typically deleted once the file has been closed
sparse	yes			the file is masquerading as a massive file with large areas of nothing
compressed	yes	yes		the file is compressed on disk
offline	yes			the file content is unavailable right now, it is stored somewhere else

Method	**GetFileDetails**			
	indexed	yes		the file content will be used in Windows Search results
	encrypted	yes		the file content is encrypted
	virtual	yes		the file exists in a virtual space rather than physically occupying disk space
	immutable	yes	yes	the file cannot be altered
	touch access time	yes		update the access time when the file is accessed
	append only	yes		the file can only be opened for reading or appending to the end
	copy on write	yes		the original file, when written to, is copied elsewhere then the write is on that copy, preserving the original
	sync directory writes	yes		force writes of directories to disk surface
	nodump	yes	yes	skip the file when 'dump' is used to back up the file
	compression error	yes		used by the experimental compression patches to indicate that a compressed file has a compression error
	extents	yes		the file storage mechanism uses extents rather than listing every block individually, this also indicates less fragmentation of the file
	huge	yes		file storage units are in blocks rather than sectors

Method	**GetFileDetails**		
	htree indexed	yes	the htree program should index this file
	journaled	yes	file writes are written to the journal first, then to the file itself
	delete securely	yes	the file is deleted first by overwriting it on disk with zeros then deleting it
	sync updates	yes	any writes to the file are written immediately to the disk surface, not cached
	top directory	yes	indicates that subdirectories with this directory are scattered around different areas of the disk
	no tail merging	yes	see https://en.wikipedia.org/wiki/Block_suballocation
	undeletable	yes	the item cannot be deleted regardless of the other permissions available
	raw access to compression	yes	indicates that a raw contents of a compressed file can be accessed directly by the experimental compression patches
	dirty compression	yes	indicates that a raw contents of a compressed file are 'dirty'
	arch	yes	the file has been archived (opposite of windows archive)
	opaque	yes	make a file opaque, for instance a directory 'file'
	sappnd	yes	system append only file

Method	**GetFileDetails**		
	schg	yes	make the system file unchangable (immutable)
	uappnd	yes	user append only file
	uchg	yes	make the user file unchangable (immutable)

Mac Specific

On macOS from version 10.4 it is possible, through the **xattr** and **ls -l@** commands to add, edit, delete and display arbitrary extended attributes on a file system item. These will be listed, if present on the file, after any standard attributes above if bestowed on the file.

Extended attributes used by Apple include these commonly found :

- com.apple.FinderInfo
- com.apple.LaunchServices.OpenWith
- com.apple.ResourceFork
- com.apple.TextEncoding
- com.apple.genstore.info
- com.apple.genstore.orig_perms_v1
- com.apple.genstore.origdisplayname
- com.apple.genstore.origposixname
- com.apple.metadata:com_apple_backup_excludeItem
- com.apple.quarantine

Example

```
FileSystem.GetFileDetails(FilePath:"c:\path\file.txt", ComputeHash: true);
```

Platforms

- Windows
- Linux
- MacOS

Notes

Calculating the hash can be expensive so it is optional.

Some result columns may not apply to the platform that the agent is running on, for instance on Unix it is not possible to determine when a file system item was

Method	**GetFileDetails**

created. In such circumstances the row will contain empty cells for those columns that are not applicable to the platform.

This method has been restricted to only search the local computer for the translation of SIDs into "domain\user".

If the Owner/Group SID identifies a user in a domain (i.e. not local to the device), then the SID will be stringized into the "S-1-5-21-xxx-yyy" format.

The reason for this local lookup only is that the Agents involved will all be running the instruction at roughly the same time and thus this could be interpreted as a denial of service attack on the domain controllers.

Important

OneDrive can report a file creation time after the last modified time.

1.6.1.53 Http.Post

Method	**Post**
Module	Http
Library	Core
Action	Send the supplied Body text via a HTTP request to the specified Uri endpoint to consume and return the response if completed. This Method is primarily for use with Splunk Event Collectors, but it might support other similar Web APIs.
Parameters	Url (string): The absolute URL of the end point to make the request to. *`Authorization`* (string, optional): The value for an *`Authorization`* request header. *Body* (string, optional): Always a JSON encoded string to send. It is sent verbatim if *`ContentType`* is "application/json" (which is highly likely for a lot of APIs) otherwise it is unescaped from JSON to a stream of octets and sent like that. *`ContentType`* (string, optional, default "*`application/json`*"): The media type of the body supplied. A comprehensive list can be found at https://www.iana.org/assignments/media-types/media-types.xhtml. *`Headers`* (table, optional, default empty): A table of two columns *`[Header x Value]`* with multiple rows, one for each header line in the request. Header is the

Method	**Post**

Key to the table, the last value in the table will prevail if there are duplicate key Headers.

Stagger(boolean, optional, default true): if *true* then the method will stagger using a random time period within the *DefaultStaggerRangeSeconds* setting before sending the post request, if *false* then the method will execute the post immediately.

Return values

Completed (boolean): Whether the request and response both completed. If false, the *ErrorMessage* column should be consulted.

ErrorMessage (string): if *Completed* is false then an error that will help to diagnose the problem.

If *Completed* is true then the following columns will also be in the output:

- *StatusCode* (integer): The HTTP Status Code returned in response to posting the body to the Uri specified.

If *Completed* is true then the following columns might also be in the output if the server supplied them:

- Date (string): The time on the server when the response was sent.

If *Complete* is true *and* a Body was sent in the response, then the following columns will also be output if the server supplied them:

- *Body* (string}: JSON-encoded text that represents the response from the server. If the *Content-Type* is "*application/json*" then the value is exactly as returned by the server, otherwise it is JSON-encoded (and the *Content-Length* will not match the length of this string).
- *Content-Length* (integer): The length (in octets) of the contents of the body that was sent by the server. It is not necessarily the length of the *Body* that is returned by this method.
- *Content-Type* (string): The content type of what is in the Body, e.g. "*text/html*".
- *WWW-Authenticate* (string): The authentication method used to gain access to the resource.
- *X-Powered-By* (string): E.g. "*ASP.NET*".

Note

Method	**Post**

Regardless of the success or otherwise, two ever-present columns indicating the success or otherwise of the operation are returned, '*Completed*' and '*ErrorMessage*'. If the request is successful then a plethora of unique response headers and values will be transformed into additional output columns, e.g. "*Connection*", "*Server*", "*WWW-Authenticate*" etc.

The standard HTTP headers have '-' (U+002D hyphen-minus) in them, and are supported. If the header names that the server returns have more unusual characters in them then it may cause issues with the SCALE language, however this is extremely unlikely to occur.

TIMS can complain about the hyphens in some of the column names.

Example **Splunk Event Collector Post all-in-one**

```
@response = Http.Post(Url : "https://input-prd-p-ftgwgk5k7h7p.clou
d.splunk.com:8088/services/collector/event",
                      Authorization : "Splunk xxxxxxxx-xxxx-xxxx-xx
xx-xxxxxxxxxxxx",
                      Body : "{\"sourcetype\" : \"my_test_data\", \
"event\": \"Simple event: HEC token in Authorization header\"}");
```

Splunk Event Collector Post using a table parameter for Headers

```
@headers = SELECT 'Authorization' AS [Header], 'Splunk xxxxxxxx-xx
xx-xxxx-xxxx-xxxxxxxxxxxx' AS [Value];

@response = Http.Post(  Url : "https://input-prd-p-ftgwgk5k7h7p.cl
oud.splunk.com:8088/services/collector/event",
                        Headers: @headers,
                        Body : "{\"sourcetype\" : \"my_test_data\
", \"event\": \"Simple event: HEC token in Authorization header\"}
");
```

A more complex example shows constructing a json object to post by using a table:

Using a table to build a Json body (assuming the server will accept this format)

```
@eventTable = SELECT "my_test_data" as sourcetype, "Simple event:
HEC token in Authorization header" as event;

@eventJson = Utilities.JsonFromTable(Table:@eventTable);

// @eventJson.Json is '[{"event":"Simple event: HEC token in Autho
rization header","sourcetype":"my_test_data"}]'
// that is an array but maybe the server doesn't accept arrays
```

```
@eventJson = SELECT SUBSTR(@eventJson.Json, 2, LENGTH(@eventJson.J
son)-2) AS Json FROM @eventJson;

// @eventJson.Json is '{"event":"Simple event: HEC token in Author
ization header","sourcetype":"my_test_data"}'
// now it's just a single object.
// Do not REPLACE(REPLACE(@eventJson.Json, '[', ''), ']', '') beca
use that could change the content rather than the markup

@response = Http.Post(Url : "https://input-prd-p-ftgwgk5k7h7p.clou
d.splunk.com:8088/services/collector/event",
                      Authorization : "Splunk xxxxxxxx-xxxx-xxxx-xx
xx-xxxxxxxxxxxx",
                      Body : @eventJson.Json);
```

And how to examine the contents of the response object returned from Splunk. Utilities.TableFromJsonrequires a Json Array [..] of Objects {..} as input.

Manually building Json objects

```
@response = Http.Post(Url : "https://input-prd-p-ftgwgk5k7h7p.clou
d.splunk.com:8088/services/collector/event",
                      Authorization : "Splunk xxxxxxxx-xxxx-xxxx-xx
xx-xxxxxxxxxxxx",
                      Body : "{\"sourcetype\" : \"my_test_data\", \
"event\": \"Simple event: HEC token in Authorization header\"}");

// @response.Body is '{"text":"Success","code":0}' this is a singl
e object, a table is an array of homogeneous objects.

@bodyJson = SELECT "[" || Body || "]" AS Json FROM @response;

// wrap array notation around the object to make a json representa
tion of a SCALE table

@bodyTable = Utilities.TableFromJson(Json:@bodyJson.Json);

// @bodyTable is :
//
// code | text
// -----+-------------
// 0    | Success
```

Extra care has to be taken handling Json and binary data. The output Body column is ALWAYS JSON. If the server returned Json then it is exactly what the server returned. If the server returned something else, including binary data, then Body

Method	**Post**

	will contain the JSON encoded version, so that it can be processed in SCALE. But this creates difficulties in handling the data.
	
	Handling binary data in the Body parameter / column
	
	`@binaryBody = SELECT CHAR(1) || CHAR(2) AS BinaryInUtf8;`
	`// @binaryBody.BinaryInUtf8 is '1 2'`
	
	`@jsonBody = Utilities.JsonFromTable(Table:@binaryBody);`
	`// @jsonBody.Json is '[{"BinaryInUtf8":"\u0001\u0002"}]'`
	
	`@jsonBinary = SELECT SUBSTR(@jsonBody.Json, 18, LENGTH(@jsonBody.Json) - 19) AS BinaryInJson FROM @jsonBody;`
	`// @jsonBinary.BinaryInJson is '"\u0001\u0002"'`
	
	`@jsonManual = SELECT "[{\"Binary\":\"\\u0001\\u0002\"}]" AS Json;`
	`// @jsonManual.Json is '[{"Binary":"\u0001\u0002"}]'`
	
	`@table = Utilities.TableFromJson(Json:@jsonManual);`
	`// @table.Binary is '1 2'`

Platforms	• Windows • Linux • MacOS
Notes	This method is primarily for use with Splunk Event Collectors, but it might support other similar Web APIs. Web APIs vary wildly, so it crucial to understand exactly what the API call requires and how that can be constructed using the body field.

1.6.1.54 Interaction.GetActiveResponses

Method	**GetActiveResponses**
Library	Core
Action	List all active survey responses.
Parameters	(None)
Return values	*FreeText* (string): Free text value input by the user.

Method	**GetActiveResponses**
	This value is empty if a free text field was not offered as part of the survey, or if the user never entered a value for this field.
Response (string): Text of the survey choice selected by the user.	
ResponseTime (string): Time when the user responded to the survey, dismissed the survey or it timed out.	
ResponseType (string): Indicates whether the user responded, dismissed the survey or the survey timed out.	
ResponseValue (float): The value of the response chosen by the user. The highest ordered choice (first in the table of choices) will have a response value of 1. The lowest ordered choice (last in the table of choices) will have a response value of 0. Those in between will be allocated a fractional value between 1 and 0 relative to their position. For example, if the choices are: (Great, Good, Average, Poor, Terrible) these will be scored (Great=1.0, Good=0.75, Average=0.5, Poor=0.25, Terrible=0)	
SurveyName (string): Name of the survey.	
UserName (string): Username of the logon session associated with this survey response.	
ValidityTime (string): The time when this survey is no longer valid.	
Example	`@responses = Interaction.GetActiveResponses();` `SELECT SurveyName, UserName FROM @res`

Method	**GetActiveResponses**
	`ponses WHERE ResponseType = "timed ou t";`
Platforms	• Windows
Notes	For more details on what is considered an active or valid response, see Interaction.ShowSurvey.

1.6.1.55 Interaction.GetInteractionState

Method	**GetInteractionState**
Library	Core
Action	Display the User Interaction state for all user sessions.
Parameters	(None)
Return values	*DoNotDisturb* (boolean): Indicates whether this user session has been set to Do Not Disturb mode. *FocusedExecutable* (string): The executable image filename for the process that is in the foreground. This may be empty if there is no active foreground window. *LastInteractionTime* (string): The last time that input was received for this user session. Typically the last time mouse or keyboard input was received. *LastNotificationShown* (string): The last time that the 1E Client displayed a notification or pop-up for this user session. *LastStateChangeTime* (string): The last time that a session state change was detected, such as screen lock or unlock. *LogonTime* (string): The logon time for this session. *NotificationState* (string): Current Notification state, which will be one of: • *Available* - Notifications can be freely sent.

Method	**GetInteractionState**
	• *NotPresent* - A screen saver is displayed, the machine is locked, Do Not Disturb has been set or a non-active Fast User Switching session is in progress. • *Busy* - A full-screen application is running or Presentation Settings are applied. Presentation Settings allow a user to put their machine into a state fit for an uninterrupted presentation, such as a set of PowerPoint slides. • *FullScreen* - An application is running in full-screen (exclusive) mode. • *Presenting* - An application has entered presentation mode, which will suppress notifications and pop-ups. • *QuietTime* - The user is in their first hour of activity, having logged into this device for the first time. During this time, notifications should not be sent or shown (to allow the user to get used to this device) *SessionId* (integer): Id of the logon session for this user. *StartupTime* (string): Start up time of the 1E Client UI in this session. *UserName* (string): Username of the logon session.

Example	`@interaction = Interaction.GetInteractionState();`
	
	`SELECT "Last interaction from " || UserName || " was " || CAST((JULIANDAY(DATETIME("now")) - JULIANDAY(DATETIME(LastInteractionTime))) * 1440 AS INTEGER) || " minutes ago" AS LastActivity FROM @interaction;`

Platforms	• Windows
Notes	Use of this method may have user privacy implications (please observe local regulations).

1.6.1.56 Interaction.GetResponseHistory

Method	**GetResponseHistory**
Library	Core
Action	Get a response history of User Interaction surveys.

Method	**GetResponseHistory**
Parameters	(None)
Return values	*FreeText* (string): Free text value input by the user. This value is empty if there was no free text field as part of the survey, or if the user never entered a value for this field. *Invalidated* (boolean): Indicates if the survey was invalidated for this user session. *Response* (string): Text of the survey choice selected by the user. *ResponseTime* (string): Time when the user responded to the survey, dismissed the survey or it timed out. *ResponseType* (string): Indicates whether the user responded, dismissed the survey or the survey timed out. *ResponseValue* (float): The value of the response chosen by the user. The highest ordered choice (first in the table of choices) will have a response value of 1. The lowest ordered choice (last in the table of choices) will have a response value of 0. Those in between will be allocated a fractional value between 1 and 0 relative to their position. For example, if the choices are: (Great, Good, Average, Poor, Terrible) then these will be scored: Great=1.0, Good=0.75, Average=0.5, Poor=0.25, Terrible=0. *SurveyName* (string): Name of the survey. *UserName* (string): Username of the logon session associated with this survey response. *ValidityTime* (string): The time when this survey is no longer valid.

Example	`@surveyResponses = Interaction.GetResponseHistory();`
	`SELECT FreeText, Response, ResponseValue, SurveyName, UserName FROM @surveyResponses WHERE`
	`  ResponseType = "responded"`

Method	**GetResponseHistory**
	`AND Invalidated = false` `AND JULIANDAY(ValidityTime) > JULIANDAY(DATETIME("now"));`
Platforms	• Windows
Notes	For more details on what is considered an active or valid response, see Interaction.ShowSurvey.

1.6.1.57 Interaction.GetSessions

Method	**GetSessions**
Library	Core
Action	Get a list of user sessions.
Parameters	(None)
Return values	*Port* (integer): Port in use by the User Interaction UI *Processid* (integer): Process Id of the User Interaction UI *SessionId* (integer): Id associated with this session. *UserName* (string): Username of this logon session.
Example	`Interaction.GetSessions();`
Platforms	• Windows
Notes	

1.6.1.58 Interaction.GetWindows

Method	**GetWindows**
Library	Core
Action	List all open windows within user sessions.
Parameters	(None)

Method	**GetWindows**
Return values	*HasFocus* (boolean): Indicates whether this window currently has screen focus. *Process* (integer): Process name *Processid* (integer): Process id *SessionId* (integer): Id associated with this session. *UserName* (string): Username of this logon session. *WindowState* (string): State of the window: Normal, Maximized or Minimized. *WindowTitle* (string): Title of the window.

Example	`@windows = Interaction.GetWindows();`
	`@edge = SELECT 1 FROM @windows WHERE Process = "MicrosoftEdge";`
	`IF (@edge)`
	`  @chrome = SELECT 1 FROM @windows WHERE Process = "chrome";`
	`  IF (@chrome)`
	`  Interaction.ShowQuestion(Async:false,`
	`  Title:"Browser survey",`
	`  Description:"Do you prefer Microsoft Edge or Google Chrome?",`
	`  PositiveChoice:"Microsoft Edge",`
	`  NegativeChoice: "Google Chrome");`
	`  ENDIF;`
	`ENDIF;`

Platforms	• Windows
Notes	Use of this method may have user privacy implications (please observe local regulations).

1.6.1.59 Interaction.InvalidateResponse

Method	**InvalidateResponse**
Library	Core
Action	Invalidate a User Interaction survey response.
Parameters	*UserName* (string, optional): Username of the logon session associated with the response to be invalidated.

Method	**InvalidateResponse**
	SurveyName (string, optional): Survey name associated with the response to be invalidated. At least one of *UserName* or *SurveyName* must be given. If a *UserName* is not supplied then the invalidation will be directed at all active responses that match the supplied *SurveyName*, and if a *SurveyName* is not supplied then the invalidation will be directed at all active responses that match the supplied *UserName*.
Return values	(None)
Example	`Interaction.InvalidateResponse(UserName:"itdomain\\john.smith", SurveyName:"UserExperienceSurvey_HR_Portal_EaseOfUse");`
Platforms	• Windows
Notes	

1.6.1.60 Interaction.SetDoNotDisturb

Method	**SetDoNotDisturb**
Library	Core
Action	Set or unset User Interaction Do Not Disturb mode for a user session.
Parameters	*Minutes* (integer): Duration of Do Not Disturb in minutes. A value of 0 can be used to unset Do Not Disturb mode.
Return values	*DoNotDisturb* (integer): How long in minutes Do Not Disturb mode will remain in force. 0 minutes means Do Not Disturb mode is now disabled. *SessionId* (integer): Id associated with this user session. *UserName* (string): Username associated with this logon session.
Example	`Interaction.SetDoNotDisturb(Minutes: 10);`
Platforms	• Windows

Method	**SetDoNotDisturb**
Notes	

1.6.1.61 Interaction.ShowAnnouncement

Module	Interaction
Library	Core
Action	Displays a pop-up announcement to one or more user sessions.
Parameters	*Name* (string): Text that will appear when the pop-up announcement is displayed. *Description* (string): Description text that will appear when the announcement is displayed. Supports "\n" as line break line. *Async* (bool): Whether to run the method synchronously or asynchronously.. In synchronous mode (*false*), the method will return a table of results once all user sessions have responded or have timed out. In asynchronous mode (*true*), the method returns immediately and results will appear as events as user sessions respond or timeout. See also *Topic*. *Icon* (string, optional): Image/icon to identify who is sending the announcement. Can be a URL or Data URL. *From* (string, optional): Name and/or e-mail of who is sending the announcement. Supports "\n" as line break line.

FromDesignation (string, optional): Title or position of who is sending the announcement. Supports "\n" as line break.

Can only be specified if *From* is given.

Choices (table, optional): A table of ordered buttons to be displayed as part of this announcement.

The table consists of rows of Text, and Action strings. The Text value will be used as the button label. If the Action value is an URL, a new browser window will be opened. The order of the choices determines the right to left order of buttons.

ShowDefaultButtons (bool, optional; default *true*): Whether to show or hide default buttons ("*Later*" and "*Dismiss*").

TimeoutSeconds (integer, optional, default 60): The number of seconds before the announcement will timeout.

Minimum of 30 seconds, maximum of 900 seconds.

Topic (string): In asynchronous mode, the name of the event that will be raised when a user response is obtained. In synchronous mode, this is not relevant and should not be supplied.

UserName (string, optional): Username of the logon session to which this announcement should be directed.

If Username is not supplied then the announcement will be directed to all active user sessions.

Module	Interaction
Link (string, optional): A URL which will appear as a 'Click here to find out more' link within the announcement.	
Footer (string, optional): Extra explanatory text to display.	
HighPriority (bool, optional, default *false*): Whether to display immediately.	
Return values	*Response* (string): Text of the button selected by the user.
ResponseTimeSeconds (integer): Number of elapsed seconds before the user responded or before the request timed out.	
ResponseType (string): Indicates whether the user responded or whether the user failed to respond before the announcement timed out.	
ResponseValue (bool): *true* if the positive choice was selected else *false* if for the negative choice.	
SessionId (integer): Id of the user session associated with this announcement response.	
UserName (string): Username of the logon session associated with this announcement response.	

Example	`@choices = SELECT "I'll be there" AS Text, "go" AS Action`
	`UNION ALL`
	`SELECT "No thanks" AS Text, "nogo" AS Action;`
	`Interaction.ShowAnnouncement(Async: false, Name:"Restaurant suggestion", Description: "Eat at Joe's", Choices: @choices, From: "Big Joe", Footer: "Bring a bottle");`

Tip

Module	Interaction
	The order of @choices determines the right to left order of buttons. In the above example, UNION ALL causes the set to stay in order written, and not be sorted by SQL.
Platforms	• Windows
Notes	

1.6.1.62 Interaction.ShowNotification

Method	**ShowNotification**
Library	Core
Action	Displays a pop-up notification to one or more user sessions. The notification includes a title, a description and the type of icon to be displayed. The method will return immediately with an indication of whether the user session was successfully notified. Notifications do not require a user response.
Parameters	*`Description`* (string): Description text that will appear when the notification is displayed. *`TimeoutSeconds`* (integer, optional, default 60): The number of seconds before the notification will timeout. Minimum of 30 seconds, maximum of 900 seconds. *`Title`* (string): Title text that will appear when the notification is displayed. *`Type`* (string, optional, default "*`Information`*"): Type of notification, which will be represented by an icon within the notification. It may be one of: *`Information`*, *`Warning`* or *`Error`*. *`UserName`* (string, optional): Username used to identify the session to which this notification should be directed.

Method	**ShowNotification**
	If Username is not supplied then the notification will be directed to all active user sessions.
Return values	*Result* (string): An indication of whether this user session was successfully notified. A string value of Notified meaning successfully notified. *SessionId* (integer): Id of the user session that was notified. *UserName* (string): Username of the logon session that was notified.

Example	`Interaction.ShowNotification(Title:"Scheduled maintenance", Description:"Documentation system going down in 5 minutes", Type: "Information", TimeoutSeconds:300);`
	
	`@description = SELECT "For security reasons, you are not permitte d to use Zoom on your company PC. " ||`
	`char(13) || "Please use either:" || char(13) || "Microsoft Teams, Cisco Webex, Google Hangout ";`
	`Interaction.ShowNotification(Description:@description, Title:"Col laboration Tools Policy",Type:"Warning");`

Platforms	• Windows
Notes	

1.6.1.63 Interaction.ShowQuestion

Method	**ShowQuestion**
Library	Core
Action	Display a pop-up question to one or more user sessions. The pop-up includes a title, a description and positive and negative choice buttons.
Parameters	*Async* (boolean): Whether to run the method synchronously or asynchronously. In synchronous mode, the method will return a table of results once all user sessions have responded or have timed out.

Method	**ShowQuestion**
	In asynchronous mode, the method returns immediately and results will appear as events as user sessions respond or timeout. *Description* (string): Description text that will appear when the pop-up question is displayed. *NegativeChoice* (string, optional, default "*No*"): Text for the negative choice button. *PositiveChoice* (string, optional, default "*Yes*"): Text for the positive choice button. *TimeoutSeconds* (integer, optional, default 60): The number of seconds before the question will timeout. Minimum 30 seconds, maximum 900 seconds. *Title* (string): Title text that will appear when the pop-up question is displayed. *Topic* (string): In asynchronous mode, the name of the event that will be raised when a user response is obtained. In synchronous mode, Topic is not relevant and should not be supplied. *UserName* (string, optional): Username of the session to which this pop-up question should be directed. If Username is not supplied then the pop-up question will be directed at all active user sessions.
Return values	*Response* (string): Text of the button selected by the user. *ResponseTimeSeconds* (integer): Number of elapsed seconds before the user responded or before the request timed out. *ResponseType* (string): Indicates whether the user responded or whether the user failed to respond before the pop-up question timed out. *ResponseValue* (boolean): *true* if the positive choice was selected else *false* for the negative choice. *SessionId* (integer): Id of the session associated with this question response. *UserName* (string): Username of the logon session associated with this question response.

Method	**ShowQuestion**

Example	`// Synchronous question to a single user session.`
	`Interaction.ShowQuestion(Async:false, Description:"Are you happy with your machine performance?",`
	` NegativeChoice:"No, not really", PositiveChoice:"Yes, thanks", Title:"System performance",`
	`UserName:"acme\\development");`
	
	`// Asynchronous question to all user sessions on this device.`
	`Interaction.ShowQuestion(Async:true, Description:"Are you happy with the network speed?",`
	`Title:"network performance", Topic:"Network question");`

Platforms	• Windows
Notes	

1.6.1.64 Interaction.ShowSurvey

Method	**ShowSurvey**
Library	Core
Action	Display a pop-up survey to one or more user sessions. The pop-up includes questions text followed by a set of selectable choices, and an optional free text box.
Parameters	*AllowFreeText* (boolean, optional, default *false*): Indicates whether a text box should be included with the survey to allow input of free text comments. If set to *true* and free text is supplied in the user response, then it is returned in the *FreeText* field.

Async (boolean): Whether to run the method synchronously or asynchronously.

In synchronous mode the method will only return once all user sessions have responded or have timed out.

In asynchronous mode, the method returns immediately and results will appear as events as user sessions respond or timeout.

Choices (string): A table of ordered choices to be displayed as part of this pop-up survey.

The table consists of rows of Text strings, and optionally a Color value. An Icon may also be optionally associated with each entry.

The Color value can be either a RGB hex code such as '#bf930f' or a color description such as 'blue'.

The Icon can be specified based on the list shown in the notes section below.

The order of the choices determines the response value that is returned when a choice is made; for details see *ResponseValue* below.

Description(string): Description text that will appear when the pop-up question is displayed. Max 512 characters.

FreeTextLabel (string, optional, default "*Anything more you'd like to tell us?*"): If *AllowFreeText* is *true*, then this can be used to set an alternative label for the free text field of the survey.

Link (string, optional): A URL which will appear as a 'Click here to find out more' link within the survey.

Name (string): Name of the survey.

This is used to uniquely identify the survey, and prevent repeat showing of the same survey.

Method	**ShowSurvey**

Question (string): Question text that will be displayed as part of this survey. Max 128 characters.

ResponseValidityMinutes (integer, optional, default 0): A response validity period in minutes.

The maximum value allowed is 129600 minutes (90 days). Default is 0 which means survey maximum validity period of 90 days will apply.

Response validity is a way to avoid repeatedly presenting a user with the same survey. When a user provides an answer to a survey, the response they give is deemed to be "valid" for the duration specified here. You may then use the Interaction.GetActiveResponsesmethod to test whether there is already a valid answer for a particular user/survey; if there is, you may wish to avoid calling Interaction.ShowSurvey and instead use their existing answer.

Once the response validity period has elapsed, the response is no longer marked as valid, and will not appear in the table returned by Interaction.GetActiveResponses.

You can force the one or more responses to be invalidated for a user/survey immediately by using the Interaction.InvalidateResponse method.

TimeoutSeconds (integer, optional, default 60): The number of seconds before the survey will timeout.

Minimum value is 30 seconds, maximum value is 900 seconds.

Topic (string): In asynchronous mode, this must be supplied and is the name of the event that will be raised when a user response is obtained.

In synchronous mode this is not relevant and should not be supplied.

UserName (string, optional): Username of the session to which this survey should be directed. If Username is not

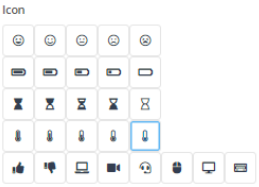
Method	**ShowSurvey**
supplied then the survey will be directed at all active user sessions.	
Return values	*FreeText* (string): Free text that was supplied as part of the user response. *Response* (string): Text of the selected choice. *ResponseType* (string): Indicates whether the user responded or failed to respond before the survey timed out. Values are: "dismissed", "responded", "snoozed", "timed out", "invalid". *ResponseValue* (double): The value of the response chosen by the user. The highest ordered choice (first in the table of choices) will have a response value of 1. The lowest ordered choice (last in the table of choices) will have a response value of 0. Those in between will be allocated a fractional value between 1 and 0 relative to their position. For example, if the choices are: (Great, Good, Average, Poor, Terrible) then these will be scored: Great=1.0, Good=0.75, Average=0.5, Poor=0.25, Terrible=0. *UserName* (string): Username of the logon session associated with this response.

Example	`@a = SELECT "Very Happy" AS Text, "green" AS Color, "far-fa-grin-squint" AS Icon`
	`UNION ALL`
	`SELECT "Happy" AS Text, "#bf930f" AS Color, "far-fa-smile" as Icon`
	`UNION ALL`
	`SELECT "Indifferent"`

Method	**ShowSurvey**

	AS Text, "blue" AS Color, "far-fa-meh" as Icon
	UNION ALL
	SELECT "Quite unhappy " AS Text, "orange" AS Color, "far-fa-frown" as Icon
	UNION ALL
	SELECT "Very Unhappy" AS Text, "red" AS Color, "far-fa-tired" as Icon;
	
	// Synchronous survey to a single user session.
	Interaction.ShowSurvey(Async:false, Choices : @a, Name:"User Experience Survey",
	Question : "How happy are you with your user experience?",
	UserName:"acme\\development", AllowFreeText: true, FreeTextLabel: "Please elaborate");
	
	// Asynchronous survey to all user sessions on this device and include a free text box.
	Interaction.ShowSurvey(Async:true, Choices : @a, Name:"User Experience Survey",
	Question : "How happy are you with your user experience?",
	AllowFreeText: true, FreeTextLabel: "Please elaborate", Topic:"MyExperienceSurvey");

Platforms	• Windows

Method	**ShowSurvey**
Notes	 In order of appearance, these are: far-fa-grin-squint.png far-fa-smile.png far-fa-meh.png far-fa-tired.png far-fa-frown.png fas-fa-battery-full.png fas-fa-battery-three-quarters.png fas-fa-battery-half.png fas-fa-battery-quarter.png fas-fa-battery-empty.png far-fa-hourglass.png fas-fa-hourglass-start.png fas-fa-hourglass-half.png fas-fa-hourglass-end.png far-fa-hourglass-o.png fas-fa-thermometer-full.png fas-fa-thermometer-three-quarters.png fas-fa-thermometer-half.png fas-fa-thermometer-quarter.png

Method	**ShowSurvey**
	fas-fa-thermometer-empty.png fas-fa-thumbs-up.png fas-fa-thumbs-down.png fas-fa-laptop.png fas-fa-video.png fas-fa-headset.png fas-fa-mouse.png fas-fa-desktop.png fas-fa-keyboard.png

1.6.1.65 Inventory.EvaluateItem

Method	**EvaluateItem**
Module	Inventory
Library	Inventory
Action	Signal extended inventory item for evaluation.
Parameters	*Name* (string): Name of the extended inventory item. *ForceFullReplication* (boolean): Force a full replication.
Return values	*Result* (string): Result of the operation. *Message* (string): Inventory item status message.
Example	`Inventory.EvaluateItem(Name:"Software", ForceFullReplication:true);`
Platforms	• Windows • Linux • MacOS

Method	**EvaluateItem**
Notes	• This method marks the named extended inventory item as pending for the next evaluation cycle. • When this method is not forcing a full replication the data will be evaluated according to the existing replication strategy.

1.6.1.66 Inventory.GetLocalReplica

Method	**GetLocalReplica**
Module	Inventory
Library	Inventory
Action	Get the local replica table for an extended inventory item
Parameters	*Name* (string): Name of the extended inventory item.
Return values	*Result* (DataTable): The local replica table for the extended inventory item if it exists.
Example	`Inventory.GetLocalReplica(Name:"Software");`
Platforms	• Windows • Linux • MacOS
Notes	The local replica table is only stored for extended inventory items with the 'Delta' replication strategy.

1.6.1.67 Inventory.GetReplicationAudit

Method	**GetReplicationAudit**
Module	Inventory
Library	Inventory
Action	Get the replication audit table for an extended inventory item.

Method	**GetReplicationAudit**
Parameters	Name (string): Name of the extended inventory item.
Return values	*Result* (DataTable): The replication audit table for the extended inventory item if it exists
Example	`Inventory.GetReplicationAudit(Name:"Software");`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.68 Inventory.GetReplicationState

Method	**GetReplicationState**
Module	Inventory
Library	Inventory
Action	Get the extended inventory replication state table.
Parameters	
Return values	*Result* (DataTable): The extended inventory replication state table if it exists.
Example	`Inventory.GetReplicationState();`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.69 Inventory.GetTables

Method	**GetTables**
Module	Inventory
Library	Inventory
Action	Returns details about the $tables used by Tachyon Activity Record.
Parameters	(None)
Return values	*Table* (string): The name of the table (without the $ prefix) *RowCount* (int): The number of rows in the table.

Example	`@tables = Inventory.GetTables();`
	`@hasDnsLive = SELECT 1 FROM @tables WHERE Table = "DNS_Live" COLLATE NOCASE;`
	`IF (@hasDnsLive)`
	`  // perform query on table`
	`ELSE`
	`  // take appropriate action`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	This method is useful to perform a defensive query against the inventory tables (which may not exist if the specific capture source has been disabled) For more information, see Tachyon Activity Record.

1.6.1.70 NativeServices.GetWmiNames

Method	**GetWmiNames**
Module	NativeServices
Library	Core
Action	Where they exist, returns all the namespaces, classes and columns that match the locator strings.

Method	**GetWmiNames**
Parameters	*Namespace* (string): The WMI namespace to match. *Class* (string): The WMI class to match within the above WMI namespace. *Column* (string): The column name to match within the above WMI Class.
Return values	If nothing matches all three parameters then the instruction will return success but no data. If anything matches then the result set will contain one row for each match on the three locator strings. • *Namespace* (string): A WMI namespace that matched. • *Class* (string): A WMI class within the above WMI namespace that matched. • *Column* (string): A column from the above WMI class that matched
Example	This should match one row on all Windows machines `@result = NativeServices.GetWmiNames(Namespace:"root\\cimv2", Class:"Win32_Processor", Column:"Manufacturer");`
Platforms	• Windows

Notes	The parameters can be wildcard expressions, so this method can dump every namespace, class and column if desired:
	
	`@result = NativeServices.GetWmiNames(Namespace:"*", Class:"*", Column:"*");`
	
	which would be a big result set with many thousands of rows. Or it could be narrowed down:
	
	`@result = NativeServices.GetWmiNames(Namespace:"root\\*", Class:"Win32_P*", Column:"M*");`
	
	For all namespaces beginning "root\\", for all the classes in those namespaces starting with "Win32_P", for all the columns in those classes starting with M, return the matching results will be issued as one unique WMI Namespace + class + column row per match in the result set.
	
	Be warned that the last example presented here took 85 seconds to complete and returned 603 rows, so try to be specific if possible.

1.6.1.71 NativeServices.RegistryDeleteKey

Method	**RegistryDeleteKey**
Module	NativeServices
Library	Core
Action	Delete a registry key.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case: Long Name / Short Name HKEY_CLASSES_ROOT / HKCR HKEY_CURRENT_CONFIG / HKCC HKEY_CURRENT_USER / HKCU HKEY_LOCAL_MACHINE / HKLM *Subkey* (string): The registry key to look for.
Return values	*Status* (string) • If the key was successfully deleted, set to "*Deleted key*". • If the key does not exist, nothing is returned, i.e. success no content. • If the **parent** key does not exist, set to (slightly misleadingly) "*Key does not exist*". If some other registry error occurs, the method fails with a descriptive message.
Example	`NativeServices.RegistryDeleteKey(Hive:"HKLM", Subkey:"software\\1e\\test\\test2");`
Platforms	• Windows
Notes	To delete keys in the *HKEY_USERS* hive, use NativeServices.RegistryDeleteUserKey.

1.6.1.72 NativeServices.RegistryDeleteUserKey

Method	**RegistryDeleteUserKey**
Module	NativeServices
Library	Core
Action	Deletes a given key and all subkeys and values for every user under HKU.
Parameters	*Subkey* (string): The registry key to look for.
Return values	For each user for whom the key existed: *Status* (string): The deletion status of the key. "*Deleted key*" if it has been successfully deleted. *Sid* (string): SID the key was found under. *Username* (string): Domain\Username of the aforementioned SID. No result is returned for users for whom the key did not exist.
Example	`NativeServices.RegistryDeleteUserKey(Subkey:"somethingSpecific");`
Platforms	• Windows
Notes	Unlike most registry methods, the **8192 limit** is *not* imposed because doing so would leave the system in an inconsistent state (some users would have the key deleted, others not), and in practice it is highly unlikely that any system would have that many users. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). **Warning** This method may attempt to call AD to translate a SID to a username. No output is returned for users for whom the key does not exist, i.e. only keys that were actually deleted are reported. For consistency with other registry user methods, this would be better named as "RegistryDeleteUserKeys".

1.6.1.73 NativeServices.RegistryDeleteUserValues

Method	**RegistryDeleteUserValues**
Library	Core
Action	Deletes a given value for every user under HKU.
Parameters	*Subkey* (string): The registry key to look for. *Name* (string): The name of the registry value that requires deletion.
Return values	For each user: *Status* (string): The deletion status of the value. "*Deleted value*" if it has been deleted, else "*Key does not exist*" if there is no such containing key. *Sid* (string): SID the value was found under. *Username* (string): Domain\Username of the aforementioned SID. No result is returned for users that did not have the value (so it could not be deleted) but did have the parent key. **Note** Prior to v4.2 the "*Key does not exist*" status was "*Value does not exist*" which was true but not as accurate.
Example	`NativeServices.RegistryDeleteUserValues(Subkey:"somethingSpecific", Name:"value12");`
Platforms	• Windows
Notes	This will delete the value for every user. Use with caution! Unlike most registry methods, the **8192 limit** is *not* imposed because doing so would leave the system in an inconsistent state (some users would have the value deleted, others not), and in practice it is highly unlikely that any system would have that many users. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). **Warning** This method may attempt to call AD to translate a SID to a username.

Method	**RegistryDeleteUserValues**
	No output is returned for users for whom the value does not exist, i.e. only values that were actually deleted are reported.

1.6.1.74 NativeServices.RegistryDeleteValue

Method	**RegistryDeleteValue**
Module	NativeServices
Library	Core
Action	Delete a registry value.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM

Subkey (string): The registry key to look for.

Name (string): The name of the registry value to delete under the key.

Return values

Status (string): The deletion status of the value. "*Deleted value*" if it has been deleted, else "*Key does not exist*" if there is no such containing key.

No result is returned if the value did not exist (so it could not be deleted) but the parent key did.

> **Note**
>
> Prior to v4.2 the "*Key does not exist*" status was "*Value does not exist*" which was true but not as accurate.

Example

```
NativeServices.RegistryDeleteValue(Hive:"HKLM", Subkey:"software\\vendor\\test\\test2", Name:"failedTestValue");
```

Method	**RegistryDeleteValue**
Platforms	• Windows
Notes	To delete a registry value for every user under HKU, use NativeServices.RegistryDeleteUserValues.

1.6.1.75 NativeServices.RegistryEnumerateKeys

Method	**RegistryEnumerateKeys**
Module	NativeServices
Library	Core
Action	Enumerate the subkeys under a registry hive & key.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU

Subkey (string): The registry key to look for.

Recursive (boolean; optional, default *false*): Whether to descend recursively or not.

Return values

If *Hive* does not exist then the result is InvalidParameter and this string:

- Invalid registry hive name: *<Hive>*

If *Subkey* has invalid structure then the result is *ExecutionError* and this string:

Method	**RegistryEnumerateKeys**
	• Invalid syntax If *SubKey* exists but cannot be opened by the agent (for any reason) then it is treated as though it does not exist. If*Subkey*does not exist then the result is*SuccessNoContent*. Otherwise the results issued will have these columns for each subkey found: • *Name* (string): The full path of the a (sub)key found below the *Subkey* parameter supplied • *DefaultValue* (string): The string value assigned to the (sub)key if set (the value named *"(Default)"* in RegEdit), or *"(value not set)"* if not (like RegEdit).
Example	`NativeServices.RegistryEnumerateKeys(Hive:"HKLM", Subkey:"software \\vendor", Recursive:true);`
Platforms	• Windows
Notes	The supplied key will appear in the results if it exists, regardless of whether or not it has child keys. REG_MULTI_SZ values are displayed with a '|' between each string. There is a **limit** of **8192** subkeys being returned (plus the key itself, if it exists). This is currently not changeable.

1.6.1.76 NativeServices.RegistryEnumerateUserKeys

Method	**RegistryEnumerateUserKeys**
Module	NativeServices
Library	Core
Action	Enumerate the subkeys under a registry key for each user in the HKEY_USERS hive.
Parameters	*Subkey* (string): The registry key to look for. *Recursive* (boolean; optional, default *false*): Whether to descend recursively or not.

Method	**RegistryEnumerateUserKeys**
Return values	If *Subkey* has invalid structure then the result is *ExecutionError* and the string "*Invalid syntax*". If *SubKey* exists under HKU but cannot be opened by the agent for any user (for any reason) then it is treated as though it does not exist. If *Subkey* does not exist for any user then the result is *SuccessNoContent*. Otherwise the results issued will have these columns: • *Name* (string): the full path of a key found below the *Subkey* parameter supplied • *DefaultValue* (string): The string value assigned to the key if set (the value named "*(Default)*" in RegEdit), or "*(value not set)*" if not (like RegEdit). • *Sid* (string): The Security Identifier of the user. • *Username* (string): The Windows account that the SID relates to.
Example	`NativeServices.RegistryEnumerateUserKeys(Subkey:"software\\vendor", Recursive:true);`
Platforms	• Windows
Notes	REG_MULTI_SZ values are displayed with a '|' between each string. There is a **limit** of **8192** subkeys being returned (including the key itself for each user, if it exists). This limit applies to all users in total, *not* per user. This is currently not changeable. **Warning** This method may attempt to call AD to translate a SID to a username. The supplied key will appear in the results if it exists, regardless of whether or not it has child keys.

1.6.1.77 NativeServices.RegistryEnumerateUserValues

Method	**RegistryEnumerateUserValues**
Module	NativeServices

Method	**RegistryEnumerateUserValues**
Library	Core
Action	Get all registry values under a given key for every user under HKU.
Parameters	*Subkey* (string): The registry key to look for. All values both directly under this key and all its subkeys are fetched.
Return values	*Name* (string): Full location of the value being retrieved. *ValueType* (string): "*REG_DWORD*" etc. as appropriate. *Value* (string): The actual value of an appropriate type, converted to a string if not one already. *Sid* (string): SID the value was found under. *Username* (string): Domain\Username of the aforementioned SID.
Example	`NativeServices.RegistryEnumerateUserValues(Subkey:"Control Panel\\Accessibility\\MouseKeys");`
Platforms	• Windows
Notes	REG_MULTI_SZ values are displayed with a '|' between each string. REG_NONE keys are not supported. There is a **limit** of **8192** values being returned. This limit applies to all users in total, *not* per user. This is currently not changeable. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). Registry values of type *REG_EXPAND_SZ* are not expanded, so for example the variable with Name "*TEMP*" in the subkey "*ENVIRONMENT*" usually has a value of "*%USERPROFILE%\AppData\Local\Temp*". This is because the Tachyon Agent does not have sufficient privilege to impersonate user accounts for security reasons. **Warning** This method may attempt to call AD to translate a SID to a username.

1.6.1.78 NativeServices.RegistryEnumerateValues

Method	**RegistryEnumerateValues**
Module	NativeServices
Library	Core
Action	Enumerate the values under a registry hive and key.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM

Subkey (string): The registry key to look for.

Recursive (boolean; optional, default *false*): Whether to descend recursively or not.

Return values

For each value found:

- Name (string): The name of the value.
- *ValueType* (string): "*REG_DWORD*" etc. as appropriate.
- *Value* (string): The actual value of an appropriate type, converted to a string if not one already.

Example

```
NativeServices.RegistryEnumerateValues(Hive:"HKLM", Subkey:"softw
are\\vendor");
```

Platforms

- Windows

Notes

REG_MULTI_SZ values are displayed with a '|' between each string.

There is a **limit** of **8192** values being returned. This is currently not changeable.

Method	**RegistryEnumerateValues**
	REG_NONE keys are not supported.

1.6.1.79 NativeServices.RegistryGetUserValues

Method	**RegistryGetUserValues**
Module	NativeServices
Library	Core
Action	Fetch a registry value for every user under HKU.
Parameters	*Subkey* (string): The registry key to look for. *Name* (string): The name of the registry value for which to fetch its value. Use an empty string to get the (default) value. *UsersOnly* (bool, optional, default *false*): Get values only for keys representing "real" users (and not for keys representing system pseudo-users or groups).
Return values	*Name* (string): Full location of the value being retrieved. *ValueType* (string): "*REG_DWORD*" etc. as appropriate. *Value* (string): The actual value of an appropriate type, converted to a string if not one already. *Sid* (string): SID the value was found under.

Method	**RegistryGetUserValues**
Username (string): Domain\Username of the aforementioned SID.	
Example	`NativeServices.RegistryGetUserValues(Subkey:"Console", Name:"FaceName");`
Platforms	• Windows
Notes	REG_NONE keys are not supported. REG_MULTI_SZ values are displayed with a '|' between each string. There is a **limit** of **8192** values being returned. This limit applies to all users in total, *not* per user. This is currently not changeable. If the subkey or registry value name does not exist, then the method will return Success (no content). If searching for the (default) value and it is 'not set', then the method will return Success (no content). Empty values are returned; empty values are not the same as 'not set'. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). Registry values of type *REG_EXPAND_SZ* are not expanded, so for example the variable with Name "*TEMP*" in the subkey "*ENVIRONMENT*" usually has a value of "*%USERPROFILE%\AppData\Local\Temp*". This is because the Tachyon Agent does not have sufficient privilege to impersonate user accounts for security reasons. **Warning** This method may attempt to call AD to translate a SID to a username.

1.6.1.80 NativeServices.RegistryGetValue

Method	**RegistryGetValue**

Module: NativeServices

Library: Core

Action: Get a registry value.

Parameters: *Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU

Subkey (string): The registry key to look for.

Name (string): The name of the registry value for which to get its value.

Return values: If the specified subkey and name exist, the call will return the following:

- *Name*(string): The full path of the registry value.
- *ValueType* (string): "*REG_DWORD*" etc. as appropriate.
- *Value*(string): The actual value of an appropriate type, converted to a string if not one already.

If the specified subkey or name does not exist, the call will not return anything but does not return an error ("success, no content").

Example:

```
NativeServices.RegistryGetValue(Hive:"HKLM", Subkey:"software\\vendor", Name:"value23");
```

The *Name* returned for this is "*HKLM\software\vendor\value23*".

Method	**RegistryGetValue**
Platforms	• Windows
Notes	REG_MULTI_SZ values are displayed with a '|' between each string. REG_NONE keys are not supported. To get a registry value for every user under HKU, use NativeServices.RegistryGetUserValues.

1.6.1.81 NativeServices.RegistryKeyExists

Method	**RegistryKeyExists**
Module	NativeServices
Library	Core
Action	Test whether a registry key exists.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case: Long Name / Short Name: HKEY_CLASSES_ROOT / HKCR HKEY_CURRENT_CONFIG / HKCC HKEY_CURRENT_USER / HKCU HKEY_LOCAL_MACHINE / HKLM HKEY_USERS / HKU *Subkey* (string): The registry key to look for.
Return values	*Exists* (boolean): Whether the key exists or not.
Example	`NativeServices.RegistryKeyExists(Hive:"HKLM", Subkey:"software\\vendor\\settings");`

Method	**RegistryKeyExists**
Platforms	• Windows
Notes	

1.6.1.82 NativeServices.RegistrySetUserValues

Method	**RegistrySetUserValues**
Module	NativeServices
Library	Core
Action	Sets a registry value for every user under HKU. This will create the value and any key hierarchy required.
Parameters	*Subkey* (string): The registry key to look for. *Name* (string): The name of the registry value to set. *Value* (string): The value that will be set. *ValueType* (string): One of these supported types: "*REG_SZ*", "*REG_EXPAND_SZ*", "*REG_BINARY*", "*REG_DWORD*", "*REG_QWORD*", "*REG_MULTI_SZ*". *UsersOnly* (bool, optional, default *false*): Set values only for keys representing "real" users (and not for keys representing system pseudo-users such as Local System nor groups).
Return values	*Status* (string): The status of the key; "*Set value*" or "*Created key and set value*" as appropriate. *Sid* (string): SID the value was found under.

Method	**RegistrySetUserValues**
Username (string): Domain\Username of the aforementioned SID.	
Example	`NativeServices.RegistrySetUserValues(Subkey:"test\\test2", Name:"temporaryStringValue", Value:"no need for concern", ValueType:"REG_SZ");`
Platforms	• Windows
Notes	Unlike most registry methods, the **8192 limit** is *not* imposed because doing so would leave the system in an inconsistent state (some users would have the value, others not), and in practice it is highly unlikely that any system would have that many users. A REG_BINARY *Value* is a string of hex digits, e.g. "*546F6D*". Upper, lower or mixed case can be used. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). **Warning** This method may attempt to call AD to translate a SID to a username.

1.6.1.83 NativeServices.RegistrySetValue

Method	**RegistrySetValue**
Module	NativeServices
Library	Core
Action	Set a registry value.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Method	**RegistrySetValue**

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM

Subkey (string): The registry key to look for.

Name (string): The name of the registry value to set.

ValueType (string): One of these supported types: "*REG_SZ*", "*REG_EXPAND_SZ*", "*REG_BINARY*", "*REG_DWORD*", "*REG_QWORD*", "*REG_MULTI_SZ*".

Value (string): The actual value, always as a string.

Return values

Status (string): one of the following:

- "*Set value*" if successful and the key already existed
- "*Created key and set value*" if the key had to be created first
- a registry error message.

Example

```
NativeServices.RegistrySetValue(Hive:"HKLM", Subkey:"software\\vendor\\test", Name:"value9", ValueType:"REG_DWORD", Value:"65535");
```

```
NativeServices.RegistrySetValue(Hive:"HKLM", Subkey:"software\\1e\\test", Name:"value10", ValueType:"REG_MULTI_SZ", Value:"line1|line2|line3");
```

Platforms

- Windows

Notes

A REG_BINARY *Value* is a string of hex digits, e.g. "*546F6D*". Upper, lower or mixed case can be used.

The '|' character is used as a newline when setting REG_MULTI_SZ keys.

For setting a registry value for every user under HKU, use NativeServices.RegistrySetUserValues.

1.6.1.84 NativeServices.RegistryUserKeyExists

Method	**RegistryUserKeyExists**
Module	NativeServices
Library	Core
Action	For each user under HKU, tests whether a given key exists.
Parameters	*Subkey (string): The registry key to look for.*
Return values	For each user: *Exists* (boolean): Whether the key exists or not. *Sid* (string): SID of the user. *Username* (string): Domain\Username of the aforementioned SID.
Example	`NativeServices.RegistryUserKeyExists(Subkey:"Control Panel\\Colors");`
Platforms	• Windows
Notes	The following statement is a useful way to correlate any user account with its SID on a device. `NativeServices.RegistryUserKeyExists(Subkey:"");` There is a **limit** of **8192** values being returned. This limit applies to all users in total, *not* per user. This is currently not changeable. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). **Warning** This method may attempt to call AD to translate a SID to a username.

1.6.1.85 NativeServices.RegistryUserValueExists

Method	**RegistryUserValueExists**
Module	NativeServices

Method	**RegistryUserValueExists**
Library	Core
Action	For each user under HKU, tests whether a given value exists.
Parameters	*Subkey (string): The registry key to look for.* *Name* (string): The name of the registry value to check.
Return values	Exists (boolean): True if the value exists, false if it doesn't. *Sid* (string): SID of the user. *Username* (string): Domain\Username of the aforementioned SID.
Example	`NativeServices.RegistryUserValueExists(Subkey:"Control Panel\\Colors", Name:"Hilight");`
Platforms	• Windows
Notes	There is a **limit** of **8192** values being returned. This limit applies to all users in total, *not* per user. This is currently not changeable. If information is retrieved from the *.DEFAULT* key or a *_Classes* key, the *Username* will be reported as "*Unknown*". It is possible to determine the owner of a *_Classes* key from the SID that precedes it (which will have a correct *Username*). **Warning** This method may attempt to call AD to translate a SID to a username.

1.6.1.86 NativeServices.RunCommand

Method	**RunCommand**
Module	NativeServices
Library	Core
Action	Execute a command-line.
Parameters	*CommandLine* (string): The entire command line to run, including the CLI arguments.

Method	**RunCommand**

WorkingDirectory(string; optional): The path of the current directory when the command is executed.

SplitLines(boolean; optional, default*false*): Whether or not to return a row per line of output. Cannot be used with*InterpretAsJson*.

TrimTrailingWhitespace(boolean ; optional, default *true*): Whether or not to discard trailing whitespace characters from output row(s). New in v5.1 in a hotfix.

IncludeErrorOutput (string; optional, default "*Interleaved*" for backwards compatibility): How to handle output from the standard error stream with respect to the Output value. One of:

Interleaved	Merge output on both streams as it is produced within the limits of buffering, i.e. mimic the same behavior as before.
Auto	Return only standard output unless there is none whatsoever in which case return standard error.

Method	RunCommand

Before	Standard error comes before standard output (with an extra new-line in between to separate).
After	Standard error comes after standard output (with an extra new-line in between to separate).
Never	Only ever return standard output.
Only	Only ever return standard error (i.e. throw away standard output).
OnFailure	Return just standard error if and only if the exit code is non-zero, otherwise return just standard output.

Note

Case is not significant.

Warning

This parameter is implemented for Windows only; it is allowed but ignored for other platforms.

Method	**RunCommand**

InterpretAsJson (boolean; optional, default*false*): If specified and has value *true*, assume that the output of the command is JSON and deserialize it. The return value is then the table produced. Cannot be used with*SplitLines*.

TimeoutSecs (integer; optional, default *3600*): If the command does not terminate within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal.

TimeoutKill (boolean; optional, default *false*): If a timeout has been reached, attempt to kill the command execution sub-process. The method terminates even if killing the command fails for some reason, i.e. the method never hangs. Do not specify if there is no timeout.

> **Warning**
>
> Try to avoid using *TimeoutKill* if possible, and instead let the sub-process die in its own time. For example, consider the effect of killing an install partway through.

IgnoreExitCode(boolean, optional, default*false*): If*true*, the execution of the command will be considered successful, irrespective of the exit code returned. If*false*, any non-zero exit code will be

Method	**RunCommand**
interpreted as a failure and this method will return an error. *AsUser* (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer).	
Return values	*ExitCode* (int): The exit code from the command (or the attempt to run the command if the command is not defined). This will only be non-zero if *IgnoreExitCode is true*. **Warning** Many Windows programs set a zero exit code even on failure. *LineNumber* (int): The line number of the output, starting at 0. This is only returned if *SplitLines* is true. *Output* (string): The output that the command sends to standard output and standard error, combined. **Warning** If there is no (captured) output from the program whatsoever, the return value from this method is completely empty. There is *not* a single row with an empty *Output* value, as might be expected.

Method	**RunCommand**

Examples	**Running as the Tachyon.Agent user (LocalSystem usually)**
	
	`NativeServices.RunCommand(CommandLine:"ping 127.0.0.1", SplitLines:true, IgnoreExitCode:true);`
	
	`NativeServices.RunCommand(CommandLine:@cmdRun, SplitLines:true, WorkingDirectory:@fileDir, TimeoutSecs: 10, TimeoutKill: true);`
	
	AsUser example
	
	`@users = Users.GetLoggedOnUsers();`
	`@users = select * from @users where SessionState='Active';`
	
	`NativeServices.RunCommand(CommandLine:"c:\\windows\\notepad.exe", AsUser:@users.Account);`

Platforms	• Windows • Linux • MacOS
Notes	If the Agent is shut down while a command is running (with or without a timeout specified), the command subprocess is detached from the Agent (so that Agent shutdown is not blocked), and the Tachyon instruction's status is set to "execution failed" because Tachyon cannot know the ultimate fate of the command. It is up to the user to decide whether to resubmit the instruction or not when the Tachyon Agent resumes. By default, a long-running instruction will time out. "Long running" is 3600 seconds, i.e. 1 hour. This is to prevent a hanging command from completely blocking the Tachyon Agent, because the Agent has an exclusivity feature for safety such that even if the Agent runs with multiple workers then executing a command prevents any other worker from running an instruction in parallel. (This is because instructions that follow the command might depend on that command having run to completion.) A command hanging indefinitely would completely block

Method	**RunCommand**
	the Agent, including any instructions to kill the hanging command's process. If a command really must be run to completion, set *TimeoutSecs* to 0 to indicate no timeout. Because the Tachyon Agent has multiple workers and so can run a number of instructions in parallel, it is possible that conflicting commands could run at the same time, e.g. each attempting to install software. (Software packages can only usually be installed one at a time.) If this is likely to be a problem, the Agent's "*Exclusivity*" setting (in its configuration file) should be set to *true* which will ensure that only one *RunCommand* can run at any time across all workers, causing other *RunCommand* instances to be temporarily suspended. By default *Exclusivity= false*. *Exclusivity* was introduced in a 1E Client 4.1 hotfix; prior to then *RunCommand* instances would block, equivalent to *Exclusivity=true*.

1.6.1.87 NativeServices.RunWmiInstanceMethod

Method	**RunWmiInstanceMethod**
Module	NativeServices
Library	Core
Action	Run a WMI object method.
Parameters	*Namespace* (string; optional, default *"root\cimv2"*): The WMI namespace to use, e.g. "*root\ccm*". *Query* (string): WMI query to select the object(s) to run the method on. *Method* (string): The name of the WMI method to run. *AsUser* (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this

Method	**RunWmiInstanceMethod**
parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer). **Note** For *AsUser* to work with TIMS/TAIL/Loadgen there must either be a *1E.Client.exe* in the same directory as the TIMS/TAIL/Loadgen executable or an installed 1E Client. **Available from v8.0**. *AllInstances* (boolean): Whether to run the method on all object instances (if there is more than one) or just the first. Additional parameters for the specific method call. Their names are prefixed with "*Param_*".	
Return values	The accumulated output, if any, from the method invocation(s).
Example	`NativeServices.RunWmiInstanceMethod(Namespace:"root\\cimv2", Query:"SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'", AllInstances:true, Method:"Terminate", Param_Reason:"1");`
Platforms	• Windows
Notes	Unlike the recommendation described for the RunWmiQuery method, "*SELECT* *" must usually be specified in the query otherwise the WMI objects produced by the filter will be incomplete and the WMI

Method	**RunWmiInstanceMethod**
	method will fail if it requires objects to work on.

1.6.1.88 NativeServices.RunWmiQuery

Method	**RunWmiQuery**
Module	NativeServices
Library	Core
Action	Execure a WMI query.
Parameters	*Namespace* (string; optional, default *"root\cimv2"*): The WMI namespace to use.

Query (string): The query expression.

AsUser (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer).

> **Note**
>
> For *AsUser* to work with TIMS/TAIL/Loadgen there must either be a *1E.Client.exe* in the same directory as the TIMS/TAIL/Loadgen executable or an installed 1E Client.

ResultsAsText (boolean; optional, default *false*):

Method	**RunWmiQuery**
• *false* - return a column for each property returned by the WMI query. Null values are reported as empty string. • *true* - return a single column named "*Data*" containing a comma-delimited list of all *PropertyName:Value* returned by the WMI query. Null values are reported as*null*. **Note** *true* allows execution of WMI queries and interpretation (or at least display) of the results without the instruction having to know the table schema (WMI property names) up-front. *IgnoreErrors* (boolean; optional, default false): If true and an error occurs during execution, the method reports success. New from 1E platform v24.5.	
Return values	A row for each instance returned by the WMI query. The number of columns depends on ResultsAsText: • *false* (default) - return a column for each property returned by the WMI query. Null values are reported as empty string. • *true* - return a single column named "*Data*" containing a comma-delimited list of all *PropertyName:Value* returned by the WMI query. Null values are reported as *null*.
Example	`// return multiple columns` `@query = "SELECT AdapterType, Name, Index, InterfaceIndex, MACAddress FROM Win32_NetworkAdapter WHERE PhysicalAdapter=true";`

Method	**RunWmiQuery**

	`NativeServices.RunWmiQuery(Namespace:"root\\cimv2", Query:@query);`
	
	`//return one column 'Data'`
	`@query = "SELECT AdapterType, Name, Index, InterfaceIndex, MACAddress FROM Win32_NetworkAdapter WHERE PhysicalAdapter=true";`
	`NativeServices.RunWmiQuery(Namespace:"root\\cimv2", Query:@query, ResultsAsText:true);`

Platforms	• Windows
Notes	Avoid using "*SELECT * FROM...*" in your WMI query for instructions, especially when used as the last statement in an instruction. Instead, you should *SELECT* specific columns, which you would use in the instruction schema. This is because: • WMI queries may return a different number of columns depending on circumstances, which may cause a conflict with the instruction output schema • WMI structure may change in future • Returning specific columns is more efficient as the Agent will need to process less data • Specifying column names is easier to understand.

1.6.1.89 NativeServices.RunWmiStaticMethod

Module	NativeServices
Library	Core

Module	NativeServices
Action	Run a WMI class method.
Parameters	*Namespace* (string; optional, default *"root\cimv2"*): The WMI namespace to use, e.g. "*root\ccm*". *Class* (string): The class, e.g. "*SMS_Client*". *Method* (string): The name of the method to run. *AsUser* (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer). **Note** For *AsUser* to work with TIMS/TAIL/Loadgen there must either be a *1E.Client.exe* in the same directory as the TIMS/TAIL/Loadgen executable or an installed 1E Client. Additional parameters for the specific method call. Their names are prefixed with "*Param_*".
Return values	The output, if any, from the method.
Example	`NativeServices.RunWmiStaticMethod(Namespace:"root\\cimv2", Class:"Win32_Process", Method:"Create", Param_Commandline:"notepad.exe");`

Module	NativeServices
Platforms	• Windows
Notes	

1.6.1.90 NativeServices.RegistryValueExists

Method	**RegistryValueExists**
Module	NativeServices
Library	Core
Action	Test whether a registry value exists.
Parameters	*Hive* (string): The registry hive which has to match one of the following long or short names, ignoring case:

Long Name	Short Name
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU

Subkey (string): The registry key to look for.

Name (string): The name of the registry value to look for under the key.

Return values	*Exists* (boolean): Whether the value exists or not.
Example	`NativeServices.RegistryValueExists(Hive:"HKLM", Subkey:"software\\vendor\\test", Name:"value9");`
Platforms	• Windows

Method	**RegistryValueExists**
Notes	

1.6.1.91 Network.GetArpEntries

Method	**GetArpEntries**
Library	Core
Action	Get current list of known MAC address to IP address mappings from the network adapter table and the ARP cache.
Parameters	(None)
Return values	For each MAC Address: • *AdapterType* (string): One of "*Ethernet*", "*Token ring*", "*PPP*", "*ATM*", "*Wireless*" or "*Unknown*". For local addresses only; empty for non-local. • *AssignedBy* (string): The IP address of the DHCP server which assigned the address, if applicable. For local addresses only; empty for non-local. • *IpAddress* (string): The corresponding IP address • *IsLocal* (bool): *true* if the mapping is from the network adapter table, else *false* if from the ARP cache • *MacAddress* (*string*): The MAC address in IEEE 802 EUI-48 notation, eg. 88-a2-5e-cc-84-80 • *Subnet* (string): The IP addresses subnet in CIDR format eg 192.168.11.0/24
Example	`Network.GetArpEntries();`
Platforms	• Windows
Notes	The following are filtered out: • Loopback adapter(s) • Empty (00-00-00-00-00-00) and broadcast (FF-FF-FF-FF-FF-FF) MAC addresses

Method	**GetArpEntries**

- Multicast addresses (224.0.0.0 to 239.255.255.255)

The method name is misleading because not just the ARP cache is used.

1.6.1.92 Network.GetConnections

Method	**GetConnections**
Library	Core
Action	Get current established TCP connections, for IPv4 and also IPv6 if available.
Parameters	(None)
Return values	For each connection: • *`Class`* (string): "*`IPv4`*" or "*`IPv6`*". • *`LatencyMs`* (int): The smoothed round trip time in milliseconds from sending a packet to getting a response for this connection. • *`LocalAddress`* (string): The IP address on this host. • *`LocalPort`* (int): The port on this host. • *`PacketLossPercent`* (int): The amount of packets lost or discarded vs the number of packets sent and received successfully as a percentage. • *`ProcessId`* (int): The PID of the process associated with the connection. • *`ProcessName`* (string): The name of the process associated with the connection. • *`RemoteAddress`* (string): The IP address on the remote host for an active outgoing connection; null if only listening. • *`RemotePort`* (int): The port on the remote host for an active outgoing connection; null if only listening.
Example	`Network.GetConnections();`
Platforms	• Windows • Linux

Method	**GetConnections**
	• MacOS
Notes	Use *`Network.GetListeningEndpoints`* for the ports which are listening for incoming connections. *`LatencyMs`* and *`PacketLossPercent`* are null on every platform but Windows. Even on Windows*`PacketLossPercent`* is always null as this information is never non-zero, even in Windows' own Network Resource Monitor. (The performance counter information is not gathered, presumably for performance reasons.)

1.6.1.93 Network.GetDnsEntriesByAddress

Method	**GetDnsEntriesByAddress**
Module	Network
Library	Core
Action	Get the contents of the DNS client cache.
Parameters	*`Collapse`* (boolean): If there are multiple DNS entries for a hostname, whether to return just the first (*`true`*) or all of them (*`false`*). This is false by default.
Return values	For each host found in the DNS cache: • *`Address`* (string): The IP address. • *`HostName`* (string): The DNS name.
Example	`Network.GetDnsEntriesByAddress(Collapse:true);`
Platforms	• Windows • MacOS
Notes	

1.6.1.94 Network.GetIpAddress (deprecated)

Method	**GetIpAddress(deprecated)**
Module	Network

Method	**GetIpAddress(deprecated)**
Library	Core
Action	Gets IP addresses assigned to the machine.
Parameters	(None)
Return values	*IpAddress* (string): The IP address of the adapter *IpVersion* (string): IPv4/IPv6 *MacAddress* (string): Physical address. *AdapterName* (string): Description of the adapter the IP address is assigned to. *AdapterType* (string): Type of the adapter. For Windows, 'friendly' strings returned are based off IP_ADAPTER_INFO type defines. *SubnetMask* (string): Subnet mask for the IP address. This is only supplied for IPv4 addresses. It is implicit in an IPv6 address.
Example	`Network.GetIpAddress();`
Platforms	• Windows
Notes	The method name is misleading because more than one address may be returned. This method has been deprecated since v3.1. Use GetIpAddresses instead.

1.6.1.95 Network.GetIpAddresses

Method	**GetIpAddresses**
Module	Network
Library	Core
Action	Gets IP addresses assigned to the machine.
Parameters	(None)

Method	**GetIpAddresses**
Return values	*IpAddress* (string): The IP address of the adapter *IpVersion* (string): IPv4/IPv6 *MacAddress* (string): Physical address. This is empty if there is no hardware MAC address assigned to the adapter, for example on a loopback address. *AdapterName* (string): Friendly name of the adapter to which the IP address is assigned. *AdapterType* (string): Type of the adapter. For Windows, 'friendly' strings returned are based off IP_ADAPTER_INFO type defines. If available, this will describe the adapter hardware type. If it is unknown, it will report "Unknown: x" where x is a number corresponding to the device type returned by the operating system. These enumerations can differ between operating systems, though they are typically based off defines in RFC1573. If just "Unknown" is returned, the operating system has returned the enumeration for "Other/Unknown". *SubnetMask* (string): Subnet mask for the IP address. This is only supplied for IPv4 addresses. It is implicit in an IPv6 address. *DefaultGateway* (string): The IP address of the default network gateway. *Description* (string): Description of the network adapter. *DhcpEnabled* (bool): Indicates if DHCP is enabled. *DhcpLeaseExpiry* (string): DHCP lease expiry time. *DhcpServer* (string): The IP address of any configured DHCP server. *PrimaryDnsServer*(string): IP address of primary DNS Server. *RouteMetric*(string): The route metric used to compute the route preferences. Given a routing choice, the route with the lowest metric will be selected. *SecondaryDnsServers*(string): A comma-separated list of IP addresses of any secondary DNS servers. *HardwareInterface*(bool): The Adapter is an actual physical hardware interface or not.
Example	`Network.GetIpAddresses();`
Platforms	• Windows

Method	**GetIpAddresses**
	• Linux • MacOS
Notes	*DefaultGateway, DhcpEnabled, DhcpLeaseExpiry, DhcpServer, PrimaryDnsServer and SecondaryDnsServers* are null on every platform but Windows and MacOS. *Description, RouteMetric, HardwareInterface* are only available on Windows.

1.6.1.96 Network.GetListeningEndpoints

Module	Network
Library	Core
Action	Get the list of TCP and UDP endpoints which are listening for incoming connections, for IPv4 and also IPv6 if available.
Parameters	(None)
Return values	For each connection... • *Class* (*string*): "*IPv4*" or "*IPv6*" for TCP, or "*UDPv4*" or "*UDPv6*" for UDP. • *LocalAddress* (string): The IP address on this host. • *LocalPort* (int): The port on this host. • *ProcessId* (integer): The PID of the process associated with the connection. • *ProcessName* (string): The name of the process associated with the connection.
Example	`Network.GetListeningEndpoints();`
Platforms	• Windows • Linux • MacOS

Module	Network
Notes	Use *`Network.GetConnections`* for established connections.

1.6.1.97 Network.GetWiFiNetworks

Method	**GetWiFiNetworks**
Library	Core
Action	Returns a list of visible WiFi networks.
Parameters	Scan (boolean): If *`true`*, perform a WiFi network scan on each adapter; if *`false`*, results are returned from the operating system cache.
Return values	*`Adapter`* (string): The name of the WiFi network adapter to which this network is visible. *`SSID`* (string): The Service Set Identifier (SSID) of the network, or empty if the network is hidden. *`Authentication`* (string): The default authentication mechanism supported by the network. *`Encryption`* (string): The default encryption mechanism supported by the network. *`IsSecure`* (boolean): True if the WiFi network is secure; false if the network is open. *`IsHidden`* (boolean): True if the WiFi network is not broadcasting a public SSID. *`IsConnected`* (boolean): True if the corresponding adapter is presently connected to this network. *`Signal`* (integer): Quality of the WiFi signal; from 0 to 100 (worst to best). *`BSSIDs`* (string): For Windows, this is a comma-separated list of the Basic Service Set Identifiers (BSSIDs) supporting this network. For macOS, this is always an empty list.
Example	`Network.GetWiFiNetworks(Scan: true);`
Platforms	• Windows • MacOS **(New in v8.2)**

Method	**GetWiFiNetworks**
Notes	Calling this method with *false* for the *Scan* parameter will not trigger the network list to be refreshed; it will instead be fetched from the operating system's cache. Calling this method with *true* for the *Scan* parameter *will* trigger a WiFi network refresh on all enabled WiFi adapters. This process takes approximately 5 seconds to locate WiFi networks on Windows. If multiple WiFi network adapters are present, it is likely that they will each see a similar list of networks. This method will show each network as seen by each network adapter. On macOS, the BSSID list for each network is empty because BSSID is not accessible to the 1E Client.

1.6.1.98 Network.Wakeup

Method	**Wakeup**
Library	Core
Action	Send a Wake-on-LAN (WOL) request to a dormant device.
Parameters	*MacAddress* (string) The MAC Address to attempt to wake up
Return values	*SentTo* (string) The IP addresses that the wakeup request was sent to.
Example	`Network.Wakeup(MacAddress:"00:11:22:33:44:55");`
Platforms	• Windows. • Linux. • macOS.
Notes	Warnings: • The destination MAC address device might wake up when requested, but there is no guarantee. • The device must have WOL enabled (in BIOS, Network driver, and OS)

Method	**Wakeup**
	Considerations: • Performs network activity **Important** Network.Wakeup is only supported on macOS and Linux from 1E platform v24.7 onward. It is supported on Windows for all platform versions.

1.6.1.99 Nomad.CachedPkgStatus

Method	**CachedPkgStatus**
Module	Nomad
Library	1E
Action	Get the percentage of the specified content in the local Nomad cache.
Parameters	*`ContentId`* (string): The ID of the content to query.
Return values	*`Percent`* (float): The percentage of the content cached on the named device. *`ContentVersion`* (int): The version of the content on the named device.
Example	What percentage and version of content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c" is in the local Nomad cache, if any? `Nomad.CachedPkgStatus(ContentId : "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c");`
Platforms	• Windows
Notes	No data is returned if the device does not have the requested content. "Pkg" appears in the method name for historical reasons - the method works for any content in a Nomad cache. See also RunPkgStatusRequest.

1.6.1.100 Nomad.CancelJob

Method	**CancelJob**
Library	1E
Action	Cancels download job with the given ID and removes it from Nomad's job queue.
Parameters	*JobId* (string): The ID of the job to cancel.
Return values	
Example	What percentage and version of content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c" is in the local Nomad cache, if any? `Nomad.CancelJob(jobId: "02F1D368-5B56-47EC-A426-812CB8CA0E6D");`
Platforms	• Windows
Notes	No data is returned if the device does not have the requested content.

1.6.1.101 Nomad.FindBestPkgCache

Method	**FindBestPkgCache**
Library	1E
Action	Find the device that has the greatest percentage of the specified content in its Nomad cache.
Parameters	*ContentId* (string): The ID of the content to search for.
Return values	*MachineName* (string): The hostname (not FQDN) of the device that has the most content, or one such device if more than one have the maximum available content. *Percent* (float): The percentage of the content cached on the named device. *ContentVersion* (int): The version of the content on the named device.
Example	What device, if any, has the greatest percentage of content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c"?

Method	**FindBestPkgCache**
	`Nomad.FindBestPkgCache(ContentId : "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c");`
Platforms	• Windows
Notes	No data is returned if no device has the requested content. "Pkg" appears in the method name for historical reasons - the method works for any content in a Nomad cache.

1.6.1.102 Nomad.GetCachedPkgDetails

Method	**GetCachedPkgDetails**
Library	1E
Action	Gets details about a package cached in Nomad.
Parameters	*`ContentId`* (string): The ID of the package to query.
Return values	*`Version`* (int): Version of the package. Returned as 0 if package does not exist in Nomad cache. *`Percent`* (float): Percentage of the package currently cached. Returned as 0.0 if package does not exist in Nomad cache. *`Format`* (int): The format in which the package is stored on disk. Has one of the following four values: 0 – Original format. Package is stored as a regular folder in an unencrypted form. 1 – Compressed format. Package is stored as a compressed archive. 2 – Encrypted format. Package is stored as a compressed and encrypted archive.

Method	**GetCachedPkgDetails**
3 – CloudDp format. Package is stored in an encrypted format used by Azure. This format is typically used for packages downloaded from Azure Cloud DP. Returned as 0 (Original format) if package does not exist in Nomad cache. *AltCachePath*(string): "Alternate cache path" of the package, if applicable. Alternate cache path is a disk location different from Nomad's standard cache path where the package has been downloaded. By default Nomad always downloads into the standard cache, but an alternate path can optionally be specified at the time of queuing the download job. Returned as a blank string if package does not exist in Nomad cache or it was not downloaded at an alternate path. *CachePriority* (int): The priority of this package among all packages in Nomad cache. Value ranges from 1 to 9 with smaller number denoting higher priority. Cache priority is used by Nomad to decide which packages to delete when running out of disk space. The lower the priority, the higher the chance of being deleted. Returned as 0 if package does not exist in Nomad cache.	
Example	Get details of the cached package "PRI0000A": `Nomad.GetCachedPkgDetails (contentId: "PRI0000A")`
Platforms	• Windows
Notes	No data is returned if no DPs have the requested content.

1.6.1.103 Nomad.GetCachePath

Method	**GetCachePath**
Library	1E
Action	Gets the path on disk where Nomad's cache is located.

Method	**GetCachePath**
Parameters	None
Return values	For each DP that has the content (maybe none): *CachePath* (string): Absolute path of Nomad's cache.
Example	`Nomad.GetCachePath();`
Platforms	• Windows
Notes	

1.6.1.104 Nomad.GetCmContentLocations

Method	**GetCmContentLocations**
Module	Nomad
Library	1E
Action	Find the Configuration Manager locations (on distribution points - DPs) that have the specified content.
Parameters	*ContentId* (string): The ID of the content to query. *Version* (int): The version of the content to query.
Return values	For each DP that has the content (maybe none): *Hash* (string): The hash of the content. (The same value for all rows returned.) *Path* (string): The URL of the content. *SiteLocality* (int): The locality of the ConfigMgr site. *0* for unknown locality, *1* for fallback, *2* for remote, or *3* for local. *SiteType* (int): The type of ConfigMgr site. *0* for unknown, *1* for server, *2* for Azure. *IsSmb* (bool): *false* if the Path is an HTTP or HTTPS URL, else *true* if an SMB share.

Method	**GetCmContentLocations**
Example	What ConfigMgr distribution points, if any, have version 1 of content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c"? `Nomad.GetCmContentLocations(ContentId : "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c", Version : 1);`
Platforms	• Windows
Notes	No data is returned if no DPs have the requested content.

1.6.1.105 Nomad.GetVariable

Method	**GetVariable**
Library	1E
Action	Gets the value of the given Nomad "variable".
Parameters	*`VariableName`* (string): Name of the variable to get. A variable can be any piece of info about Nomad such as a registry setting or even some computed value based on the current state of Nomad. This API is meant to be a flexible way of getting info out from Nomad which can be extended in future without breaking any COM interfaces. Currently the following three are the only supported variables: 1. CacheAvailableBytes: The disk space currently available to Nomad for its cache in bytes. 2. CacheUsedBytes: The disk space currently used by Nomad. 3. SSDEnabled: The value of registry setting "SSDEnabled".
Return values	*`Value`* (integer): The value of the requested variable. The type depends on the variable in question, although the currently supported variables are all numeric.
Example	Get the current value of Nomad setting "SSDEnabled": `Nomad.GetVariable(VariableName: "SSDEnabled");`
Platforms	• Windows

Method	**GetVariable**
Notes	

1.6.1.106 Nomad.JobsStatus

Method	**JobsStatus**
Module	Nomad
Library	1E
Action	List active Nomad jobs.
Parameters	(None)
Return values	For each job in the queue (maybe none): *ContentId* (string): The unique identifier for the content that the job is transferring. *Version* (int): The version of the content. *Status* (string): "*Running*" or "*Paused*".
Example	`Nomad.JobsStatus();`
Platforms	• Windows
Notes	

1.6.1.107 Nomad.Pause

Method	**Pause**
Module	Nomad
Library	1E
Action	Pause any currently-executing Nomad job. Has no effect if no job is currently running.
Parameters	(None)

Method	**Pause**
Return values	*Success* (boolean): *true* if the request was successfully sent to Nomad, else *false* if Nomad is not installed, is not running, or is not a supported version.
Example	`Nomad.Pause();`
Platforms	• Windows
Notes	Because of the way Nomad works, this action sends a fire-and-forget request to Nomad. The return value reflects whether Nomad was able to accept the request, not necessarily whether Nomad acted on it. It is not an error if no Nomad job is currently running to be paused.

1.6.1.108 Nomad.PauseJobQueue

Method	**PauseJobQueue**
Module	Nomad
Library	1E
Action	Pause the Nomad job queue so that no jobs will run.
Parameters	*PauseDurationMinutes* (int; optional, default 1440): For how long to pause the Nomad job queue, in minutes. After this interval Nomad will automatically resume the job queue, if it has not been resumed already using a ResumeJobQueue instruction. The value must be > 0. Default is 1 day. *PauseFrom* (string; optional): The absolute UTC time, in ISO 8601 format as a string, from which the pause "clock" should start. If this is not given, the start time is when the instruction is executed on the Agent. The intent is that an instruction using this method specifies the value as "*%environment:now%*" (see Environment variables). This is a special value understood by the consumer API, which plugs in the time that the instruction was created. This has the effect that even if execution of the instruction is delayed on any Agents for any significant time, then the Nomad queue will resume exactly this time + *PauseDurationMinutes* minutes on *all* Agents after the instruction was submitted, not some semi-random time dependent on when a laptop, say, is booted up. See the example below. Of course you can hard-code an ISO 8601 time, but that is rarely useful. If the time given by *PauseFrom* + *PauseDurationMinutes* has passed when the Agent executes the instruction, i.e. the resume-time has already passed, then the

Method	**PauseJobQueue**
	Agent does not pause the Nomad queue at all. This is not considered an error - the instruction has run successfully but done nothing.
Return values	*Success* (boolean): *true* if the request was successfully sent to Nomad (or if the resume time has already passed). Also if Nomad is installed (and of a sufficient version) but the service is not currently running, Tachyon itself will write the registry value that controls how long Nomad will pause for when its service is restarted; *true* or *false* is returned depending on the success of the write. If Nomad is not installed or is not a supported version then the method (and hence usually the instruction running it) fails with an execution error.
Example	The following instruction was submitted at 2017-06-27T12:00:00Z. Consider a laptop which has been off but is turned on 18 hours later. The Agent catches up with instructions from the Switch, and runs the instruction. The default pause duration for the Nomad job queue is 1 day *from when the instruction was submitted*, so the Agent on the laptop pauses Nomad's job queue for 24 - 18 = 6 hours, and the queue resumes at 2017-06-28T12:00:00Z. `Nomad.PauseJobQueue(PauseFrom : "%environment:now%");` if the instruction's execution had been delayed and ran after 2017-06-28T12:00:00Z then there would be no pause at all.
Platforms	• Windows
Notes	Because of the way Nomad works, this action sends a fire-and-forget request to Nomad. The return value reflects whether Nomad was able to accept the request, not necessarily whether Nomad acted on it. It is not an error if no Nomad job is currently running to be paused. The calculated resume time becomes irrelevant if the pause is explicitly cancelled by a subsequent ResumeJobQueue instruction.

1.6.1.109 Nomad.Resume

Method	**Resume**
Module	Nomad
Library	1E

Method	**Resume**
Action	Resume any currently-paused Nomad job. Has no effect if no job is currently paused.
Parameters	(None)
Return values	*Success* (boolean): *true* if the request was successfully sent to Nomad, else *false* if Nomad is not installed, is not running, or is not a supported version.
Example	`Nomad.Resume();`
Platforms	• Windows
Notes	Because of the way Nomad works, this action sends a fire-and-forget request to Nomad. The return value reflects whether Nomad was able to accept the request, not necessarily whether Nomad acted on it. It is not an error if no Nomad job is currently paused.

1.6.1.110 Nomad.ResumeJobQueue

Method	**ResumeJobQueue**
Module	Nomad
Library	1E
Action	Resume the currently-paused Nomad job queue. Has no effect if the job queue is not currently paused.
Parameters	(None)
Return values	*Success* (boolean): *true* if the request was successfully sent to Nomad or if Nomad is not currently paused. If Nomad is not installed or the service is not running or is not a supported version then the method (and hence usually the instruction running it) fails with an execution error.
Example	`Nomad.ResumeJobQueue();`
Platforms	• Windows

Method	**ResumeJobQueue**
Notes	Because of the way Nomad works, this action sends a fire-and-forget request to Nomad. The return value reflects whether Nomad was able to accept the request, not necessarily whether Nomad acted on it.

1.6.1.111 Nomad.RunPkgStatusRequest

Method	**RunPkgStatusRequest**
Module	Nomad
Library	1E
Action	Test whether any other SSD (single site download) device has more content than the local Nomad cache.
Parameters	*ContentId* (string): The ID of the content to query. *Version* (int): The version of the content to query. *Format* (int; optional, default *0*): How the content is stored in the cache. *0* = original Nomad format (neither compressed nor encrypted), *1* = compressed, *2* = compressed and encrypted. *DeviceIdType* (string): What the following *DeviceId* parameter represents. Either "*FQDN*" (*DeviceId* is a fully qualified domain name) or "*IPv4*" (*DeviceId* is an IPv4 address). Case is not significant. IPv6 addresses are not supported. *DeviceId* (string): Either the FQDN or the IPv4 address (see *DeviceIdType*) of the SSD site master. *Percent* (float): The minimum desired percentage of content in the cache.
Return values	*Percent* (float): If there is more content on any other SSD device, the percentage which is available through SSD, which will be greater than the input *Percent* value. if no SSD device has a greater percentage than that specified, the same value as in the input *Percent* is returned. The input *Percent* value is also returned on error, e.g. if the *DeviceId* parameter is an unknown device.
Example	Does an SSD device have more than 50% of (ConfigMgr) content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c" at version 1, neither compressed nor encrypted, using BIGNOMADCACHESERVER.ACME.LOCAL as the SSD site master?

Method	**RunPkgStatusRequest**
	`Nomad.RunPkgStatusRequest(ContentId : "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c", Version : 1, Format : 0, DeviceIdType : "FQDN", DeviceId : "BIGNOMADCACHESERVER.ACME.LOCAL", Percent : 50.0` If the returned *Percent* is greater than 50.0 then an SSD device does have more content than is in the local cache.
Platforms	• Windows
Notes	Please contact your 1E Account team team for more information about the Nomad COM API to which this method is a wrapper. This method is unusual in that the parameters do not match the internal Nomad provider API. *DeviceIdType* and *DeviceId* are replaced there by *PeerFqdn* and *PeerIpv4* strings, matching the Nomad COM API.

1.6.1.112 Nomad.RunSsdQuery

Method	**RunSsdQuery**
Module	Nomad
Library	1E
Action	Use Nomad's SSD (single site download) feature to find on which devices the specified content is cached.
Parameters	*ContentId* (string): The ID of the content to query. *Version* (int): The version of the content to query. *Format* (int; optional, default *0*): How the content is stored in the cache. *0* = original Nomad format (neither compressed nor encrypted), *1* = compressed, *2* = compressed and encrypted. *Percent* (float): The minimum desired percentage of content in the cache. *Extent* (int): Whether to search the entire site (*0*) or just the local subnet (*1*). *MaxDevices* (int): The maximum number of devices to return. The range is *1* to *50*.

Method	**RunSsdQuery**
	AllSubnets (bool): Whether to return results with a good distribution of subnets (*true*) or allow it does not matter if the devices are clustered in a few subnets or even just one (*false*).
Return values	For each suitable device (and there may be none): *Fqdn* (string): The fully qualified domain name of the device. *Ipv4* (string): The IPv4 address of the device. *Ipv4Subnet* (string): The IPv4 subnet of the device.
Example	Find up to 10 devices that have at least 50% of (ConfigMgr) content "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c" version 1, neither compressed nor encrypted. Try to return devices from a number of subnets. `Nomad.RunSsdQuery(ContentId : "Content_7f368e5f-4dbe-4027-ba81-c5463e372a0c", Version : 1, Format : 0, Percent : 50.0, Extent: 0, MaxDevices : 10, AllSubnets : true);`
Platforms	• Windows
Notes	Nomad SSD must be enabled of course. IPv6 addresses are not supported.

1.6.1.113 Nomad.Status

Method	**Status**
Module	Nomad
Library	1E
Action	Get the status and version of the Nomad service.
Parameters	(None)
Return values	*Running* (boolean): Whether the Nomad service is currently running or not. *Version* (string): Nomad's version number. Empty if Nomad is not installed.
Example	`Nomad.Status();`

Method	**Status**
Platforms	• Windows
Notes	

1.6.1.114 Nomad.Stop

Method	**Stop**
Module	Nomad
Library	1E
Action	Stop any currently-executing Nomad job. Has no effect if no job is currently running.
Parameters	(None)
Return values	*Success* (boolean): *true* if the request was successfully sent to Nomad, else *false* if Nomad is not installed, is not running, or is not a supported version.
Example	`Nomad.Stop();`
Platforms	• Windows
Notes	Because of the way Nomad works, this action sends a fire-and-forget request to Nomad. The return value reflects whether Nomad was able to accept the request, not necessarily whether Nomad acted on it. It is not an error if no Nomad job is currently running.

1.6.1.115 OperatingSystem.ControlService

Module	OperatingSystem
Library	Core
Action	Action to start, stop, etc. a service or daemon.
Parameters	*Service* (string): The service or daemon name. See Notes below. *Action* (string; optional, default empty): One (or none) of "*start*", "*stop*", "*restart*", "*pause*" or "*resume*". Case is not significant. At least one of *Action*

Module	OperatingSystem

and/or *Startup* must be specified. "*pause*" and "*resume*" are only supported for Windows.

Startup (string; optional, default empty): If a non-empty string, the service startup type, one of "*automatic*", "*delayed*", "*disabled*" or "*manual*". Case is not significant. At least one of *Action* and/or *Startup* must be specified. Only used on Windows.

IncludeDependents (boolean; optional, default *false*): If specified and *true* and the action failed because of dependencies, perform the same action on the services that depend on this service then try the service action again. Only used on Windows.

StopDependencies (boolean; optional, default *false*): Previous name for *IncludeDependents*. **Deprecated - use IncludeDependents**.

> **Note**
>
> If the *Action* is "*restart*", the dependent services are always restarted automatically regardless of *IncludeDependents* / *StopDependencies*.

Return values

Action (string): A list of the action(s) successfully performed, with an empty value indicating that the action failed.

Example

```
OperatingSystem.ControlService(Service:"MisbehaveSvc", Startup:"Disabled");
```

For backwards compatibility this example use the deprecated "*StopDependencies*" name rather than "*IncludeDependents*":

```
OperatingSystem.ControlService(Service:"MisbehaveSvc", Action:"Stop", StopDependencies:true);
```

Platforms

- Windows
- Linux
- MacOS

Notes

This cannot be used to stop, pause or disable the Tachyon and 1E Client services/daemons but start, restart and resume operations are allowed.

For Windows, the *Service* parameter is the service's name, not the service's display name. For example, to stop the 'Background Intelligent Transfer Service' service, supply 'BITS'.

Module	OperatingSystem
	For Linux systems that use *systemd* to control daemons, the *Service* parameter should be the name as known to systemd, e.g. "*cups.service*". For older Linuxes that still use */etc/init.d*, the *Service* parameter should be the name of a script in the */etc/init.d* directory, e.g. "*cups*". For Mac, the *Service* parameter is the **full path** of the daemon's plist file, e.g. "*/Library/LaunchDaemons/com.1E.Daemon.plist*". Beware, because as Apple documentation states: "Due to bugs in the previous implementation [of *launchd* and the *launchctl* command] and long-standing client expectations, ... zero is always returned." So the method will always appear to succeed on the Mac even if the specified service does not exist or some other error occurs.
Warning	Setting *Startup* parameter to 'Manual' for Tachyon and 1E Client services is allowed, but the user needs to exercise caution using this functionality as it can lead to problems with the services not starting up correctly.

1.6.1.116 OperatingSystem.GetInstalledUpdates

Method	**GetInstalledUpdates**
Module	OperatingSystem
Library	Core
Action	Get list of installed Windows updates.
Parameters	(None)
Return values	*Title* (string): Update title.
Status (string): "*OK*", and/or an error code as reported by the Windows Update component.	
InstallDate (string): If available, the installation time and date.	

Method	**GetInstalledUpdates**
Example	`OperatingSystem.GetInstalledUpdates();`
Platforms	• Windows
Notes	The list consists of all updates that have ever been installed for which the Windows Update Agent has a record. This includes updates that have been superseded and/or subsequently uninstalled, So the list is not just the *currently* installed updates. The Status column can include an error code which may need to be interpreted in the absence of a corresponding message. The list of error codes is available on Microsoft's MSDN site. Also note that an error code may also be returned in the event of a successful installation; for example, the error code 0x240006 may be present to indicate that an update did not install because it was already present on the device.

1.6.1.117 OperatingSystem.GetProcesses

Module	OperatingSystem
Library	Core
Action	Get details of executing processes.
Parameters	(None)
Return values	For each current process: • *AccountName* (string): The user account. • *AccountId* (string): The user account ID; SID for Windows or UID for *nix. • *ProcessId* (integer): The PID of this process. • *ParentProcessId* (integer): The PID of the parent of this process, if this is a child process. • *StartupTime* (string): The *StartUpTime* of the process if known, otherwise NULL.

Module	OperatingSystem
	• *CommandLine* (string): If command line (with parameters) used to invoke the process. • *Executable* (string): The binary being executed. • *MemoryUsageBytes* (string): Some measure of the physical memory being used by this process in bytes. – On Windows, this is the total of the private pages of the working set (resident pages) on Windows – On platforms with the all the necessary procfs files, this is the total bytes in the process' private resident pages, this is equivalent to the Windows measure. – On platforms not covered above then this is just the private and shared resident memory. This is slightly different to the Windows measure, resident shared pages are included.
Example	`OperatingSystem.GetProcesses();`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.118 OperatingSystem.GetServiceInfo

Module	OperatingSystem
Library	Core
Action	Retrieves a list of all services, or a single named service.
Parameters	*Name* (string, optional): The name of the single service (e.g. "winmgmt" for the WMI service) to return; omit this parameter to return data for all services. See Notes.

Return values

Name (string): The service's name.

Description (string): The descriptive name of the service.

Type (string): One of:

- File system driver
- Kernel driver
- Own process
- Per-user
- Shared process
- <empty string>

Note

'Per-user' is only reported from v5.2 onwards; it was an <empty string> for per-user services in older versions.

State (string): One of:

-
  - About to continue
  - Pausing
  - Paused
  - Running
  - Starting
  - Stopping
  - Stopped

ProcessId (int): The process identifier of the service, or 0 if the service is not running.

StartType (string): One of:

Module	OperatingSystem

-
  - Automatic
  - Boot (kernel driver only)
  - Manual
  - Disabled
  - System (kernel driver only)

ErrorControl (string): One of:

-
  - SERVICE_ERROR_CRITICAL
  - SERVICE_ERROR_IGNORE
  - SERVICE_ERROR_NORMAL
  - SERVICE_ERROR_SEVERE

BinaryPath (string): The full path of the executable that is/will be started by the service controller.

StartAccountName (string): The name of the user account under which the service will be started.

DelayStart (boolean): True if the service is set to delayed auto-start (applies to Windows Vista and up).

TriggerStart (boolean): True if the service is set to start on a trigger (applies to Windows 7 and up).

Example	`OperatingSystem.GetServiceInfo();`
Platforms	• Windows
Notes	If you specify a service name to look up, it must be the complete name (i.e. the method does not do any partial matching).

Module	OperatingSystem
	If this service does not exist, or its details cannot be retrieved, the method returns no data.

1.6.1.119 OperatingSystem.GetSummary

Method	**GetSummary**
Module	OperatingSystem
Library	Core
Action	Get information about the installed version of the operating system.
Parameters	(None)
Return values	*Caption* (string): Description of the O/S.
InstallDate (string): When the current version of the O/S was installed, in ISO 8601 format.	
SystemDriveInstallDate (string): When the O/S system drive was installed, in ISO 8601 format.	
Manufacturer (string): Publisher of the O/S.	
Version (string): Version of the O/S.	
OsArchitecture (string): The processor architecture of the operating system (e.g. x86, x64).	
OsLocale (string): The operating system default locale.	

Method	**GetSummary**
Example	`OperatingSystem.GetSummary();`
Platforms	• Windows • Linux • MacOS
Notes	If the O/S has been upgraded, the *`SystemDriveInstallDate`* value should be older than that of the original install. *`InstallDate`* represents when the current version was installed, i.e. when the O/S was last upgraded. If the O/S has never been upgraded then the values will be very similar but not necessarily identical.

1.6.1.120 OperatingSystem.GetUpdatesPendingReboot

Method	**GetUpdatesPendingReboot**
Module	OperatingSystem
Library	Core
Action	Get list of installed Windows updates that are awaiting a reboot.
Parameters	(None)
Return values	*`Title`* (string): Update title. *`Status`* (string): "*`OK`*" or empty *`InstallDate`* (string): If available, the installation time and date.
Example	`OperatingSystem.GetUpdatesPendingReboot();`
Platforms	• Windows
Notes	

1.6.1.121 OperatingSystem.GetUpdateSummary

Method	**GetUpdateSummary**
Library	Core
Action	Get status details of Windows automatic update.
Parameters	(None)
Return values	*AutoUpdatesEnabled* (boolean): Whether automatic updates are enabled or not. If not, most of the following are N/A. *AutoUpdatesMandatory* (boolean): Whether automatic updates are mandatory or not. *AutoUpdateNotificationLevel* (string): Notification level, one of • *Disabled* • *Not configured* • *Notify before download* • *Notify before installation* • *Notify at scheduled installation time* *UpdateAgentStatus* (string): Status of the service. *UpdateAgentVersion* (string): Version number of the update service agent *LastSuccessfulDetect* (string): Time of last successful detection *LastSuccessfulDownload* (string): Time of last successful download *LastSuccessfulInstall* (string): Time of last successful installation *NextDetectTime* (string): Time of next scheduled detection. *HasPendingReboot* (boolean): Whether a reboot is pending.
Example	`OperatingSystem.GetUpdateSummary();`
Platforms	• Windows
Notes	For Windows 10 and Window Server 2016 (and presumably later) the *LastSuccessfulDownload* and *NextDetectTime* values are not available, and prior to Agent v3.3 nor were *LastSuccessfulDetect* and

Method	**GetUpdateSummary**
	LastSuccessfulInstall. (Microsoft have removed some registry values from these O/Ss.)

1.6.1.122 OperatingSystem.KillProcess

Module	OperatingSystem
Library	Core
Action	Action to terminate a single running process.
Parameters	*ProcessId* (string): The (numeric) process ID of the executable to kill.
Return values	*Killed* (boolean): Whether the process was successfully killed *(true*) or not *(false)*. If a process with PID *ProcessId* did not exist, *false* is returned.
Example	`OperatingSystem.KillProcess(ProcessId:"996");`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.123 OperatingSystem.KillProcesses

Method	**KillProcesses**
Module	OperatingSystem
Library	Core
Action	Terminate a running process (or set of process instances of the same executable).
Parameters	*Executable* (string): The filename of the executable to kill. For Windows, this is the basename (e.g. "*notepad.exe*").

Method	**KillProcesses**
	For Linux it can be the basename (e.g. "*firefox*") or the full path (e.g. "*/usr/lib64/firefox/firefox*").
Return values	*Killed* (int): The number of processes successfully killed. *Failed* (int): The number of processes that could not be killed.
Example	`OperatingSystem.KillProcesses(Executable:"notepad.exe");` `OperatingSystem.KillProcesses(Executable:"/usr/lib64/firefox/firefox");`
Platforms	• Windows • Linux
Notes	

1.6.1.124 Patch.Deploy

Method	**Deploy**
Module	Patch
Library	Core
Action	Deploy (i.e. download and by default install) either all available patches or specific patches to this device.
Parameters	*Source* (string; optional; default is to deduce the source): The source of patch meta-data, typically the installation mechanism used to handle the patch(es). One of:*CAB*: A file with a `.cab` extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved.*SCCM*: System Center Configuration Manager.*WSUSL*: A local (to the enterprise) corporate Windows

Method	Deploy
	Server Update Service server.*WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com.Case is not significant. **Note** Case is not significant.
PatchSpec (string; optional, defaults to an empty string implying **all** available patches will be deployed): The patches to be actioned. **Note** For Windows, this is a comma-separated list of Knowledge Base article numbers (*without* the "KB" prefix) or the GUIDs of specific patches, or even a mixture. (See example below.) Case is not significant for the hex digits in GUIDs.	
DownloadOnly (boolean; default *false*): Whether the patch should be downloaded but not be installed if it is not already installed. **Note** DownloadOnly does not apply if the source is *SCCM*. The SCCM API does not support downloading only, whereas WSUS does.	
CabFilePath (string; default empty): The full path of the location of the *.cab* file if Source is '*CAB*'. **Note** This *must* be specified if *Source* is '*CAB*' and should not be specified for any other *Source* value. If the path is specified it must be to a local CAB-file. Shared, i.e. remote, CAB-files are not supported by Windows Update Agent.	

CheckOnline (bool; default *false*): Grant permission (or not) as to whether to go over the network or remain purely local to the host. If this is *false*, patches can only be installed if they have already been downloaded to the local cache. With SCCM this may trigger installs that have already been identified and downloaded but not yet installed (ie they are ready and waiting for the deadline). A value of *true* is incompatible with *CabFilePath*.

> **Note**
>
> Setting this to *true* may cause a lot of network traffic if many devices run this method simultaneously.
>
> For a *Source* of *WSUSR* the "network" in question is the internet, and for *WSUSL* and *SCCM* it is the LAN.

Asynchronous (bool: default *false*): If *false*, then the patch will happen inline with the instruction execution, meaning the instruction could take many minutes to complete and the Agent will be unable to respond to other patch instructions effectively. If *true* then no result set is returned and a follow up call some time later to List would be needed to determine whether the patch is now installed.

> **Note**
>
> Synchronous (*Asynchronous* = *false*) patching with source SCCM is achieved by eventing and polling WMI classes as the CM client is totally independent of Tachyon and makes its own decisions. Tachyon requests the SCCM client to do something and monitors the progress to see whether the request has started and finished. There is a good chance that the installation will complete within the ten minutes allowed for 'synchronous' completion of this method, however Tachyon may decide that CM Client has

taken too long and will report back that the timeout has occurred. The CM Client driven installation will however continue in the background though and the final status can be determined via *Patch.List* using the appropriate PatchSpec.

TimeoutSecs (int; optional, default *1800*): If the Source does not complete within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal. Available from v4.0.

Note

Thisparameterisincompatible with *Asynchronous* = *true*, which will cause the instruction to return success before theoperation has even started, letalonecompleted.

If the timeout occurs the install will continue in the background as though nothing happened but *Success* is returned from the method and for any rows in the result set the *InstallError* and *InstallResult* columns will indicate that the timeout occurred.

It will be necessary to call List to determine the new state of the device.

Return values

For each of the supplied patches or if not supplied for all available patches the following is returned:

Note

If no update information is available then the method willreturnsuccess with no rows.

PatchSpec (string): The patch identifier. For Windows, a Knowledge Base article number.

Method	**Deploy**
DownloadOnly (boolean): Whether the patch was to be just staged (*false*) or also installed (*true*), i.e. the supplied or implicit *DownloadOnly* parameter. *NeededDownload* (boolean): Whether the patch actually needed to be downloaded. *DownloadResult* (integer): The COM success (0) or error (not 0) codes related to downloading, otherwise empty if no download is attempted. *DownloadError* (string): The human readable form of the *DownloadedResult* column if a download error occurred. *NeededInstallation* (boolean): Whether the patch actually needed to be installed (*true*) or it was already installed (*false*). *InstallResult* (integer): The COM success (0) or error (not 0) codes related to installation, otherwise empty (*null*) if installation is not attempted. *InstallError*(string): The human readable form of*InstallResult*if an installation error occurred. *RebootRequired* (bool): If *true* then a reboot is required after the patch was installed, if *false* then a reboot is not required. If *DownloadOnly* was requested, indicates whether a reboot will (or may) be required following installation.	
Example	`Patch.Deploy(Source:"SCCM", PatchSpec:"2267602,3182545", DownloadOnly:true);`
Platforms	• Windows
Notes	Updates will be installed only if they do not require user input. However, any

Method	**Deploy**
	license agreements are accepted automatically. Windows may refuse to allow installations if a reboot is pending. If no update information is available then the method will return success with no rows. The method does not fail with an error if the *PatchSpec* indicates a patch (or patches) that is not available, whether the source does not have it or the patch simply does not exist. In this case a sparse response record will be returned with empty *DownloadResult* and *InstallResult* values, and the Agent log will contain a "*No updates to install*" warning.

1.6.1.125 Patch.GenerateFullInventoryEvents

Method	GenerateFullInventoryEvents
Module	Patch
Library	Core
Action	Generate Full Inventory Events
Parameters	None.
Return values	None.
Example	`Patch.GenerateFullInventoryEvents();`
Platforms	• Windows
Notes	

1.6.1.126 Patch.GetMetadata

Method	**GetMetadata**
Module	Patch
Library	Core
Action	Query metadata for a single patch or comma-separated list of patches.
Parameters	*PathSpec* (string): A single patch or comma-separated list of patches to be queried. **Note** For Windows, this is a comma-separated list of Patch GUIDs. Case is not significant for the hex digits in GUIDs.
Return values	For each update requested: *ClassificationGuid* (string): Classification GUID of the patch, as defined under https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ff357803(v=vs.85 *ClassificationText* (string): Classification text of the patch, for example 'Critical Updates'. *Description*(string): Description of the patch. *Family* (string): Product family of the patch, such as 'SQL Server'. *IsRebootRequired* (bool): Indicates whether this patch will require a system reboot

Languages (string): A comma-separated list of languages for which this patch is relevant; an empty list indicates that this patch is not language specific.

PatchIds (string): The patch identifiers.

> **Note**
>
> For Windows, Knowledge Base article number(s).

Products (string): A comma-separated list of products associated with this patch.

PublishDate (string): The last published date of the update, in ISO 8601 format, on the server that deploys the patch.

Revision (string): Revision number of the patch.

SecurityBulletinIds (string): A comma-separated list of any Security bulletin IDs addressed by the patch.

> **Note**
>
> An example entry is 'MS14-059' (Where MS14 means Microsoft 2014 for example see https://docs.microsoft.com/en-us/security-updates/securitybulletins/2014/ms14-059)

Severity (string): MSRC severity of the patch, for example 'Critical'.

SupercededIds (string): A comma-separated list of any Patch GUIDs that have been superseded by this patch.

SupportUrl (string): Support Url of the patch.

Title (string): Full title of the patch.

Vendor (string): Company that supplied the patch, for example 'Microsoft'.

Method	**GetMetadata**

Example	Find the devices and their sources where software updates are configured:
	
	`// A list of Patch Guids can also be provided:`
	`Patch.GetMetadata(PatchSpec:"ca10020a-becd-4535-b06d-e30e92f01ad6,f6c4261c-9fae-442c-9bfc-7b37ba42f806");`
	
	`// The latest Patch history entries can be searched for their metadata.`
	`@entries = SELECT * FROM [Patch.History] WHERE ValidTo=0 AND KB != "";`
	`FOREACH @entry IN @entries`
	`DO`
	`  Patch.GetMetadata(PatchSpec:@entry.UpdateId);`
	`DONE;`
	
	`// The SQLite table maintained by Patch.GetMetadata can also be viewed directly:`
	`@entries = SELECT * FROM [Patch.Metadata];`
	`FOREACH @entry IN @entries`
	`DO`
	`  @json = SELECT Metadata FROM @entry;`
	`  Utilities.TableFromJson(Json:@json);`
	`DONE;`

Platforms	• Windows
Notes	

1.6.1.127 Patch.GetStatus

Method	**GetStatus**
Module	Patch

Method	**GetStatus**
Library	Core
Action	Report how updates are configured for the device.
Parameters	*CabFilePath* (string; default empty): If Source '*CAB*' is to be searched for, the full path of the *.cab*-file. **Note** If this is not given, no '*CAB*' source will be sought. If it is given, the path must be to a local CAB-file. Shared, i.e. remote, CAB-files are not supported by Windows Update Agent.
Return values	For each source of updates found on the device: *Source*(string): Where the updates come from. One of: • *CAB*: A file with a *.cab* extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved. • *SCCM*: System Center Configuration Manager. • *WSUSL*: A local (to the enterprise) corporate Windows Server Update Service server. • *WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com.

Method	**GetStatus**

SourceId (string): The identity of the source of updates. E.g. for a '*SCCM*' source the site ID, or for WSUS the URL.

Configured(bool): Whether updates are configured from the source.

Enabled (bool): Whether updates are enabled from the source.

> **Warning**
>
> Prior to v4.2 for source *WSUSR* or *WSUSL*, an *Enabled* value of *true* may not be accurate; use Patch.TestConnection to accurately test whether the source can be accessed. From v4.2 onwards the '*Configure automatic updates*' Group Policy setting is taken into account.

LastSuccessfulInstallTime (string): The time of the last successful update installation on the device.

> **Note**
>
> The values for all sources can be the same because Windows Update (which is also used by SCCM) does not distinguish between sources when recording the last successful installation.

LastSuccessfulUpdateSearchTime (string): The time of the last successful search for updates for the device.

> **Note**
>
> The values for sources CAB, WSUSL and WSUSR are the same because Windows Update does not distinguish between sources when recording the last successful search for updates.

Example

Find the devices and their sources where software updates are configured:

```
@patchStatus = Patch.GetStatus();
```

Method	**GetStatus**
	`SELECT * FROM @patchStatus WHERE Configured = 1;`
Platforms	• Windows
Notes	

1.6.1.128 Patch.List

Method	List
Module	Patch
Library	Core
Action	List the status of patches.
Parameters	*Source* (string): The source of patch meta-data, typically the installation mechanism used to handle patch(es). One of:*CAB*: A file with a *.cab* extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved.*SCCM*: System Center Configuration Manager.*WSUSL*: A local (to the enterprise) corporate Windows Server Update Service server.*WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com.Case is not significant. **Note** If this parameter is not specified, the Tachyon Agent attempts to deduce the source as follows: 1. If a CAB-file is specified and exists (although is not necessarily valid), the source is implicitly '***CAB***'.

Method	List
	2. Else if the SCCM client is enabled for software updates, the source is implicitly ' ***SCCM*** '. 3. Else if a local Windows Software Update server is configured, the source is implicitly ' ***WSUSL*** '. 4. Else if the remote Windows Software Update Service is configured, the source is implicitly ' ***WSUSR*** '. 5. Else the source is unknown and must be specified explicitly; this causes an error. **Warning** SCCM uses WSUSL behind the scenes but it hides certain information. Specifying an explicit *Source* of "*WSUSL*" may reveal more patches that have been installed than by specifying "*SCCM*" or by leaving Tachyon to deduce SCCM. For speed of response, the SCCM option does not run a Software Updates Deployment Evaluation Cycle action, so the data returned may not be completely up to date.
PatchSpec (string; optional, default all available patches): The patches to be matched against the full list of updates. If no specific patches are supplied, all updates will be listed. **Note** For Windows, this is a comma-separated list of Knowledge Base article numbers	

(*without* the "KB" prefix) or the GUIDs of specific patches, or even a mixture.

Case is not significant for the hex digits in GUIDs.

CheckOnline (bool; default *false*): Grant permission (or not) as to whether to go over the network or remain purely local to the host. This parameter only affects *WSUSL* and *WSUSR* and will have no effect on *SCCM*, where it will always be set to false.

Warning

If *CheckOnline* is *false*, the already-installed updates are generally reported accurately but, for obvious reasons, additional updates that are available from a remote server cannot be detected.

CabFilePath (string; default empty): The full path of the location of the *.cab* file if Source is '*CAB*'.

Note

This *must* be specified if *Source* is '*CAB*' and should not be specified for any other *Source* value.

If the path is specified it must be to a local CAB-file. Shared, i.e. remote, CAB-files are not supported by Windows Update Agent.

TimeoutSecs (int; optional, default *600*): If the Source does not complete within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal. Available from v4.0.

Note

Error status is returned if the timeout occurs.

Method	List
Return values	For each patch:

PatchIds (string): Comma-separated list of identifiers associated with the update, as would be specified to the Deploy method.

> **Note**
>
> For Windows these are the associated Knowledge Base article IDs, e.g. "*2267602*".

PatchGuid (string): GUID of the patch, if available.

> **Warning**
>
> This can be null if the patch is only known from the Windows quick fix engineering (QFE) database.

Title (string): Full title of the patch.

IsDownloaded (boolean): Whether the patch has been downloaded to the device but not yet installed.

> **Note**
>
> If the patch has been successfully installed (i.e. *IsInstalled* is *true*), this will be *false* even though the patch was actually downloaded at some point. *IsDownloaded* being *true* implies that the patch is ready to install.

IsInstalled (boolean): Whether the patch has been installed on this device.

Status (string): Current status of the patch on this device. One of these:

- *Available for download* (but not yet downloaded)
- *Downloading*

Method	List

- *Downloaded* (but not yet installed)
- *Installation aborted*
- *Installation failed*
- *Installation in progress*
- *Installation not started*
- *Installation successful*
- *Installation successful pending reboot*
- *Installation successful with errors*
- *Installation successful with errors pending reboot*
- *Uninstallation aborted*
- *Uninstallation failed*
- *Uninstallation in progress*
- *Uninstallation not started*
- *Uninstallation successful*
- *Uninstallation successful pending reboot*
- *Uninstallation successful with errors*
- *Uninstallation successful with errors pending reboot*
- *Unknown*

Note

A "pending reboot" status means that the action has been carried out (install or uninstall) but it will not take effect until after the device has been rebooted.

Warning

Method	List

WSUS does not always record the most recent status. For example, a few updates can apparently be at "*Installation in progress*" after some years and device reboots. *IsInstalled* is marked as *false* for such patches.

Timestamp (string): When the patch achieved its *Status* value.

Source (string): If the patch has been installed on this device (or is available for installation), what subsystem supplied that information. One of:

- *MSI*: The Windows MSI (Windows Installer, formerly "Microsoft Installer") API; all such patches are fully installed.
- *QFE*: The installed patch is known only from the Windows "Quick Fix Engineering" database.
- *REGISTRY*: The patch is known only from the Windows Registry. Typically this applies to updates that are partially installed but require a reboot to be fully installed.
- *PandF*: Information about the installed patch was obtained from the same derivation as *Control Panel → Programs and Features → View installed updates*.

CAB: A file with a *.cab* extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved.*SCCM*: System Center Configuration Manager.*WSUSL*: A local (to the enterprise) corporate Windows Server Update Service server.*WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com.

Method	List
	Warning For a patch that requires a reboot, data can be incomplete following install but prior to reboot compared with after reboot. Prior to reboot we see just this: • *PatchIds*:null, *PatchGuid*:"*3bfc94d2-adb3-41c0-a5f0-a5d54e8fa824*", *Title*:null, *InInstalled*:*true*, *Status*:"*Installation successful pending reboot*", *Source*:"*unknown*" After reboot we get: • *PatchIds*:"*4091664*", *PatchGuid*:"*8e16108f-4a20-4231-ad07-b41f662accb0*", *Title*:"*2018-10 Update for Windows Server 2016 for x64-based Systems (KB4091664)*", *Status*:"*Installation successful*", *Source*:"*WSUSR*" The problem is that Windows does not supply enough information to the Tachyon Agent prior to reboot; even the GUIDs are inconsistent.
Example	`Patch.List(Source: "WSUSL", PatchSpec:"004226dc-aaf5-4adc-9dd5-285ce0e1ad36,3182545,1CF1F075-360E-4E95-AB8D-D6B0F4B09D9B");`
Platforms	• Windows
Notes	This method usually returns a different number of records than are displayed by *Control Panel* → *Programs and Features* → *Installed Updates* for a number of reasons, including: • *Installed Updates* can display a single line for a KB ID that has

Method	List
	multiple patches, whereas Tachyon lists all the patches. • *Installed Updates* displays a patch shared between a number of products for each of the products, whereas Tachyon lists it just once. The *IsDownloaded* and *IsInstalled* properties are not strictly necessary because the *Status* implies their values, but they are useful as summaries. Patch Success uses 4 instructions, 1 from SLA inventory and 3 from the 1E-PatchSuccess product packs. Patch Success uses this method it its **1E-PatchSuccessRefresh** instruction and then aggregates the status for each update into the following headings: • **Fully patched** WHEN IsInstalled = 1 AND Status = 'Installation successful' THEN "Installed successfully" • **Requires rebooting** WHEN IsInstalled = 1 AND Status = 'Installation successful pending reboot' THEN "Installed successfully - pending reboot" • **Requires attention** WHEN Status LIKE '%with errors%' OR Status LIKE 'Installation failed' THEN "Failed - " || Status • **Requires patching** WHEN IsInstalled = 0 THEN "Missing" • **No patches** if there are no patches reported.

Method	List
	Patch Success uses this method in its **1E-PatchSuccessExplore** instruction to return the following for each patch on a device: • **Title** • **Status** • **Source** • PatchGuid as **PatchIdent** • **Timestamp** Patch Success uses this method in its **1E-PatchSuccess-PatchStatus** instruction (from SLA inventory) to return the following for each patch on a device: • PatchGuid as **PatchIdent** • **Title** • **IsDownloaded** • **IsInstalled** • **Status** • **Timestamp** • **Source** The 4th instruction is described in Patch.Deploy.

1.6.1.129 Patch.Restart

Method	**Restart**
Module	Patch
Library	Core
Action	Microsoft-suggested procedure to fix Windows Update Agent problems, which has optional steps that include a restart of the Windows Update service.

Method	**Restart**
Parameters	*SkipDeleteQmgr*(bool; default true): Skip the step that deletes 'qmgr*.dat' on directory '%ALLUSERSPROFILE%/Application Data/Microsoft/Network/Downloader *SkipRegSvr32* (bool; default true): Skip the step that force COM registration of Patch required dlls. *SkipResetBits*(bool; default true): Skip BITS service restart. Some changes may only be visible after restart. *SkipResetWinsock*(bool; default true): Skip Winsock reset. *SkipResetWUA*(bool; default true): Skip 'wuauserv' service restart. Some changes may only be visible after restart.
Return values	For each operation performed on the device: *Description* (string): Description of any error encountered whilst performing the operation, for example 'DllRegisterServer Not Found *Name* (string): Name of the operation performed, for example 'Stopping service wuauserv' *Status* (string): Status of the operation, for example 'Ok'
Example	Fix Windows Update Agent problems `Patch.Restart(SkipResetWUA: false);`
Platforms	• Windows
Notes	See https://docs.microsoft.com/en-gb/windows/deployment/update/windows-update-resources for details.

1.6.1.130 Patch.TestConnection

Method	**TestConnection**
Module	Patch
Library	Core
Action	Check that the source for patches is accessible.

Method	**TestConnection**
Parameters	*Source* (string): The source of patch meta-data, typically the installation mechanism used to handle patch(es). One of:*CAB*: A file with a *.cab* extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved.*SCCM*: System Center Configuration Manager.*WSUSL*: A local (to the enterprise) corporate Windows Server Update Service server.*WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com.Case is not significant. **Note** If this parameter is not specified, the Tachyon Agent attempts to deduce the source as follows: 1. If a CAB-file is specified and exists (although is not necessarily valid), the source is implicitly '***CAB***'. 2. Else if the SCCM client is enabled for software updates, the source is implicitly '***SCCM***'. 3. Else if a local Windows Software Update server is configured, the source is implicitly '***WSUSL***'. 4. Else if the remote Windows Software Update Service is configured, the source is implicitly '***WSUSR***'. 5. Else the source is unknown and must be specified explicitly; this causes an error. **Note** When testing a WSUSL connection, this method includes a built-in random stagger interval so that multiple Tachyon Agents do not overwhelm the local WSUS server simultaneously.

Method	**TestConnection**
CabFilePath (string; default empty): The full path of the location of the *.cab* file if Source is '*CAB*'. **Note** This *must* be specified if *Source* is '*CAB*' and should not be specified for any other *Source* value. If the path is specified it must be to a local CAB-file. Shared, i.e. remote, CAB-files are not supported by Windows Update Agent. **Warning** Only the existence of the CAB-file is checked; the contents are NOT validated.	
Return values	*ConnectionOk* (boolean): Whether the Source can be accessed or not.
Source (string): Where the updates come from. One of: *CAB*: A file with a *.cab* extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved.*SCCM*: System Center Configuration Manager.*WSUSL*: A local (to the enterprise) corporate Windows Server Update Service server.*WSUSR*: The remote Windows Server Update Service feed over the internet at microsoft.com. **Note** This is useful if the source is not specified as an input parameter and has to be deduced.	

Method	**TestConnection**
SourceId (string): The identity of the source of updates. E.g. for a '*SCCM*' source the site ID, or for WSUS the URL.	
Example	`@ConfigMgrOk = Patch.TestConnection(Source: "SCCM")`
Platforms	• Windows
Notes	

1.6.1.131 Policy.ForceStatusReport

Method	**ForceStatusReport**
Module	Policy
Library	Core
Action	Force a report of policy rule status the next time the Agent restarts.
Parameters	None
Return values	*Message* (string): A message confirming whether the status report has been scheduled.
Example	`Policy.ForceStatusReport();`
Platforms	• Windows • Linux • MacOS
Notes	This method sets an internal flag instructing the Agent to report a full set of policy rule compliance status messages - even if the compliance status has not changed - the next time they are evaluated following a restart. The policy is NOT evaluated or reported on immediately.

1.6.1.132 Policy.GetEventSubscriptions

Method	**GetEventSubscriptions**
Module	Policy
Library	Core
Action	Get the list of event subscriptions in the current policy.
Parameters	None
Return values *`Topic`* (string): The event subscription topic. *Type* (string): A JSON block containing application-specific data for this subscription (or empty).	For each event subscription:
Example	`Policy.GetEventSubscriptions();`
Platforms	• Windows • Linux • MacOS
Notes	If the policy engine has been disabled through configuration, this method returns no data. This method will return example data (for testing) in TIMS, because TIMS uses an alternative policy engine.

1.6.1.133 Policy.GetRuleEvaluationHistory

Method	**GetRuleEvaluationHistory**
Module	Policy
Library	Core

Method	**GetRuleEvaluationHistory**
Action	Get a chronological history of recent rule evaluation (for one rule, or all rules).
Parameters	*`RuleId`* (integer, optional): The identifier of a specific rule whose history should be retrieved. Omit for a history of all rules.
Return values	For each rule evaluation: *`RuleId`* (integer): Rule identifier. *`Changed`* (bool): Whether this evaluation marks a change from the previous state of the rule. *`Status`* (integer): The numeric representation of the evaluation status (see notes). *`StatusText`* (string): The textual description of the evaluation status (see notes). *`Timestamp`* (string): The date and time corresponding to the evaluation.
Example	`Policy.GetRuleEvaluationHistory();`
Platforms	• Windows • Linux • MacOS
Notes	The data returned by this method may include evaluations for previous rules which are no longer active. The amount of data available to, and returned by, this method is determined by the data retention policy - typically approx. 5000 rows.

Method	**GetRuleEvaluationHistory**
	The mapping of status codes is as follows: 0: 'Received' 1: 'CheckErrored' 2: 'CheckFailed' 3: 'CheckPassed' 4: 'FixErrored' 5: 'FixPassed' 6: 'FixFailed' 7: 'NotApplied' (the rule has been disabled by the policy manager) 8: 'PreConditionErrored' 9: 'PreConditionPassed' 10: 'PreConditionFailed' If the policy engine has been disabled through configuration, this method returns no data. This method will return example data (for testing) in TIMS, because TIMS uses an alternative policy engine

1.6.1.134 Policy.GetRules

Method	**GetRules**
Module	Policy
Library	Core
Action	Get the list of rules, along with their status, in the current policy.
Parameters	None
Return values	For each rule:

Method	**GetRules**
RuleId (integer): The server-defined numeric identifier for the rule. *Name* (string): The name of the rule. *Triggers* (string): A JSON block containing an array of triggers and their parameters where applicable. *RuleType* (string): The type of rule, either '*Check*' or '*Fix*'. *Status* (integer): The numeric representation of the evaluation status (see notes). *StatusText* (string): The textual description of the evaluation status (see notes). *Timestamp* (string): The date and time corresponding to the rule's current status.	
Example	`Policy.GetRules();`
Platforms	• Windows • Linux • MacOS
Notes	The mapping of status codes is as follows: 0: 'Received' 1: 'CheckErrored' 2: 'CheckFailed' 3: 'CheckPassed' 4: 'FixErrored'

Method	**GetRules**
	5: 'FixPassed' 6: 'FixFailed' 7: 'NotApplied' (the rule has been disabled by the policy manager) 8: 'PreConditionErrored' 9: 'PreConditionPassed' 10: 'PreConditionFailed' If the policy engine has been disabled through configuration, this method returns no data. This method will return example data (for testing) in TIMS, because TIMS uses an alternative policy engine

1.6.1.135 Policy.GetStatistics

Method	**GetStatistics**
Module	Policy
Library	Core
Action	Returns a single row with policy-related statistics for the lifetime of the Tachyon Agent process.
Parameters	None
Return values	*`PolicyDownloads`* (integer): The number of times a policy document has been downloaded. *`PolicyBytesDownloaded`* (integer): The total number of bytes downloaded across all policy documents.

Method	**GetStatistics**
PolicyFailedDownloads (integer): The number of times a policy document failed to download. *PolicyUpdateRequests* (integer): The number of server-instigated requests for this Tachyon Agent to download a policy. *PolicyRuleEvaluations* (integer): The total number of rule evaluations. *PolicyRuleEvaluationErrors* (integer): The total number of rule evaluations which resulted in an error. *PolicySuccessfulFixes* (integer): The total number of fixes applied following a non-compliant rule. *NotificationsGenerated* (integer): The total number of policy status notifications generated. *NotificationsSent* (integer): The total number of policy status notifications sent to the server. *EventsGenerated* (integer): The total number of events generated for forwarding to the server (based on event subscriptions). *EventsSent* (integer): The total number of events sent to the server.	
Example	`Policy.GetStatistics();`
Platforms	• Windows • Linux • MacOS

Method	**GetStatistics**
Notes	As per the description, these statistics are for the lifetime of the process (and are reset if the Tachyon Agent restarts). If the policy engine has been disabled through configuration, this method returns no data. This method will return example data (for testing) in TIMS, because TIMS uses an alternative policy engine

1.6.1.136 Policy.GetSummary

Method	**GetSummary**
Module	Policy
Library	Core
Action	Get a summary of the current policy applied to this Tachyon Agent.
Parameters	None
Return values	*`EventSubscriptionCount`* (integer): The number of event subscriptions contained within the policy. *`Hash`* (string): The SHA-256 hash of the policy document content (or empty if there is no active policy). *`RuleCount`* (integer): The number of rules contained within the policy. *`Size`* (integer): The size in bytes of the policy document content. *`Timestamp`* (string): The date and time at which the policy was downloaded and applied.

Method	**GetSummary**
Example	`Policy.GetSummary();`
Platforms	• Windows • Linux • MacOS
Notes	If the policy engine has been disabled through configuration, this method returns no data. This method will return example data (for testing) in TIMS, because TIMS uses an alternative policy engine.

1.6.1.137 Policy.Invalidate

Method	**Invalidate**
Module	Policy
Library	Core
Action	Reset this Agent to have a null/empty policy. **Warning** This method should be used for troubleshooting purposes only.
Parameters	None
Return values	*`Message`* (string): A message confirming the policy invalidation.
Example	`Policy.Invalidate();`
Platforms	• Windows • Linux • MacOS
Notes	This method immediately resets the Agent's policy to an empty one, which discards all rules and event subscriptions. If the Agent is connected to the Switch, the Agent will subsequently negotiate an appropriate policy, although this may

Method	**Invalidate**
	take a few minutes to happen. If the Agent is not connected to the Switch, no policy rules or event subscriptions will be in effect, and the Agent will renegotiate a policy the next time it does connect.

1.6.1.138 Scripting.RunText

Method	**Run**
Module	Scripting
Library	Scripting
Action	Execute the specified script supplied as text.
Parameters	*Language* (string): The shell language of the script. Currently "*PowerShell*" and "*bash*" (case is not significant) are supported.

LanguageVersion (string; optional): If specified, the version of the script interpreter, e.g. "3.0" for PowerShell if features not supported in older versions are used. A runtime error occurs if the requested version of the interpreter is not available on the host.

Text (string): The text of the script.

> **Warning**
>
> The text should be in an appropriate format for the target operating system. For example, a multi-line bash script should have lines separated by just linefeed (newline) characters, not Windows' carriage-return + linefeed.

InterpretAsJson (boolean; optional, default *false*): If specified and has value *true*, assume that the output of the script is JSON and deserialize it. The return value is then the table produced.

> **Warning**

If InterpretAsJson is set to true, then the output of the script must be serialized as JSON. PowerShell 3.0 and later has a ConvertTo-Json cmdlet. If using PowerShell 2.0 then LanguageVersion must be set to 2.0 and your script will need to include a function to convert to JSON; an example is provided here _Using PowerShell commands and scripts.

TimeoutSecs (integer; optional, default *3600* : If the script does not terminate within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal.

TimeoutKill (boolean; optional, default *false*): If a timeout has been specified and is reached, attempt to kill the script execution sub-process. The method terminates even if killing the script fails for some reason, i.e. the method never hangs. Do not specify if there is no timeout.

Warning

Try to avoid using *TimeoutKill* if possible, and instead let the sub-process die in its own time. For example, consider the effect of killing an install partway through. Also, only the script interpreter is killed, not any sub-sub-processes it may invoke.

ExecutionPolicy (string; optional, default empty): **Can only be used with PowerShell scripts**. If specified, set the named PowerShell execution policy at the Process scope (i.e. as if the command "*powershell.exe -ExecutionPolicy policy ...*" was used to execute the script). This can be used to get round lower-scope execution policy settings that might otherwise block script execution. All PowerShell execution policy values are accepted. However, "*Bypass*" is probably the most useful and "*Undefined*" is pointless.

In addition to the standard execution policy values, there is an additional special " ***Override*** " value. This allows the host's execution policy to be bypassed

completely, even at the*MachinePolicy*and*UserPolicy*scopes. In other words, "*Override*" allows a PowerShell script to run *whatever* the local settings

ExecutionPolicy is not case-sensitive.

IgnoreExitCode (boolean, optional, default *false*): If *true*, the execution of the script will be considered successful, irrespective of the exit code returned. If *false*, any non-zero exit code will be interpreted as a failure and this method will return an error.

> **Note**
>
> *IgnoreExitCode* only applies if the script actually executes and terminates. This is not a general "keep going regardless of anything failing" switch. For example, if an attempt is made to run a PowerShell script but *powershell.exe* itself is broken, then execution of the method fails regardless of the *IgnoreExitCode* setting.

WorkingDirectory (string, optional): This will set the working directory for the script execution to the specified path. If the path does not exist, the 1E Client will return an error.

AsUser (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer).

script_params... (optional): Zero or more "anonymous" parameters (i.e. just values, with no " *name:*" prefix)

Method	**Run**
that will be passed directly to the script as its command line parameters. They can be of various types (string, int, etc.). The script parameters can be included anywhere in the list of method parameters, because they are anonymous.	
Return values	If parameter *InterpretAsJson* was *true*, the data table produced. Otherwise: *ExitCode* (int): The exit code from execution of the script. -2 if the script timed out. **Warning** Depending on the script, an exit code of 0 does not necessarily mean that execution was successful. *Output* (string): The text produced by the script's execution on standard output and standard error streams. *User* (string): Aligns the row output to the user that the output refers to.

Example	This generates an *Output* value of "*Hello\nGoodbye*" where *\n* is a newline character.
	
	`Scripting.RunText(Language: "Bash", Text: "echo Hello; echo Goodbye");`
	
	AsUser example
	
	`@users = Users.GetLoggedOnUsers();`
	`@allExecutions = FOREACH @user IN @users`
	`DO`
	`  @account = SELECT Account FROM @user;`
	`  Scripting.RunText(Languag`

Method	**Run**
	e: "Powershell", Text:"echo Hello; echo Goodbye", AsUser:@account); DONE; **Passing a path as the Text** It is possible to use this method to execute a pre-existing script file as the Text parameter exactly as if within a PowerShell prompt, for example: Scripting.RunText(Language:"PowerShell", Text:"C:\\ProgramData\\1E\\simpleEcho.ps1 $($args[0]) $($args[1])","hello","world"); Script arguments must be explicitly forwarded as shown, as only the author of the above line knows exactly where the parameters should go within the text field. However, the above is really what the Scripting.Run() method is designed for, which makes things far more readable.
Platforms	• Windows - PowerShell • Linux - bash • MacOS - bash
Notes	This is an alternative to the Scripting.Run method which instead uses a script in a file. Because the 1E Client has multiple workers and so can run a number of instructions in parallel, it is possible

Method	Run
	that conflicting scripts could run at the same time, e.g. each attempting to install software. (Software packages can only usually be installed one at a time.) If this is likely to be a problem, the 1E Client's "*Exclusivity*" configuration value should be set to *true* which will ensure that only one script can run at any time across all workers, causing other scripts to be temporarily suspended.By default *Exclusivity* is *false*.This method does not have a SplitLines parameter. If the script output is a single text block and not JSON, then you can use the Utilities.SplitLines method to split the text block into lines.PowerShell script parameters are in effect surrounded by single quotes when the script is executed. This is to allow parameter values to contain embedded spaces, and to allow what can be metacharacters (e.g. '*(*' for PowerShell or '*$*' for bash) to be treated as literals. This means that PowerShell script parameters cannot contain embedded single quotes. bash scripts can contains embedded quotes.If the 1E Client is shut down while a script is running (with or without a timeout specified), the script subprocess is detached from the 1E Client (so that 1E Client shutdown is not blocked), and the Tachyon instruction's status is set to "execution failed" because the 1E Client cannot know the ultimate fate of the script. It is up to the user to decide whether to resubmit the instruction or not when the 1E Client resumes.By default, a long-running

Method	**Run**
	instruction will time out. "Long running" is 3600 seconds, i.e. 1 hour. This is to prevent a hanging script from completely blocking the 1E Client, because the 1E Client has an exclusivity feature for safety such that even if it runs with multiple workers then executing a script prevents any other worker from running an instruction in parallel. (This is because instructions that follow the script's execution might depend on that script having run to completion.) A script hanging indefinitely would completely block the 1E Client, including any instructions to kill the hanging script's process. If a script really must be run to completion, set *`TimeoutSecs`* to 0 to indicate no timeout.

1.6.1.139 Scripting.Run

Module	Scripting
Library	Scripting
Action	Execute the specified script file.
Parameters	*`Language`* (string): The shell language of the script. Currently "*`PowerShell`*" and "*`bash`*" (case is not significant) are supported. *`LanguageVersion`* (string; optional): If specified, the version of the script interpreter, e.g. "3.0" for PowerShell if features not supported in older versions are used. A runtime error occurs if the requested version of the interpreter is not available on the host. *`ScriptPath`* (string): The full path of the script file.

Note

While the earlier "*Script*" name will still work, we recommend that *ScriptPath* is used instead.

Warning

The script file must be in a format suitable for the target O/S. For example, the line-end character for a bash script must be just newline, not Windows-style carriage-return + newline.

Tip

Recommended to be local. Use the httpGetFile() function to download a file. The output of the function is FilePath.

InterpretAsJson (boolean; optional, default *false*): If specified and has value *true*, assume that the output of the script is JSON and deserialize it. The return value is then the table produced.

Warning

If InterpretAsJson is set to true, then the output of the script must be serialized as JSON. PowerShell 3.0 and later has a ConvertTo-Json cmdlet. If using PowerShell 2.0 then LanguageVersion must be set to 2.0 and your script will need to include a custom function to convert to JSON.

TimeoutSecs (integer; optional, default *3600*): If the script does not terminate within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal.

TimeoutKill (boolean; optional, default *false*): If a timeout has been specified and is reached, attempt to kill the script execution sub-process. The method terminates even if killing the script fails for some reason, i.e. the method never hangs. Do not specify if there is no timeout.

Warning

Try to avoid using *TimeoutKill* if possible, and instead let the sub-process die in its own time. For example, consider the effect of killing an install partway through. Also, only the script interpreter is killed, not any sub-sub-processes it may invoke.

ExecutionPolicy (string; optional, default empty): **Can only be used with PowerShell scripts**. If specified, set the named PowerShell execution policy at the Process scope (i.e. as if the command "*powershell.exe -ExecutionPolicy policy ...*" was used to execute the script). This can be used to get round lower-scope execution policy settings that might otherwise block script execution. All PowerShell execution policy values are accepted, including " *Bypass*". However *ExecutionPolicy* provides an additional special " ***Override*** " value, which allows the host's execution policy to be bypassed completely, even at the *MachinePolicy* and *UserPolicy* scopes. In other words, "*Override*" allows a PowerShell script to run *whatever* the local settings.

ExecutionPolicy values are not case-sensitive.

IgnoreExitCode (boolean, optional, default *false*): If *true*, the execution of the script will be considered successful, irrespective of the exit code returned. If *false*, any non-zero exit code will be interpreted as a failure and this method will return an error.

Note

IgnoreExitCode only applies if the script actually executes and terminates. This is not a general "keep going regardless of anything failing" switch. For example, if an attempt is made to run a PowerShell script but *powershell.exe* itself is broken, then execution of the method fails regardless of the *IgnoreExitCode* setting.

WorkingDirectory (string, optional): This will set the working directory for the script execution to the specified path. If the path does not exist, the agent will return an error.

AsUser (string; optional; default 1E.Client service user) **Windows only**: The "domain\user" (down-level logon) account name that the 1E Client will attempt to impersonate on the local machine.If this parameter is absent the command will be executed in the 1E Client process's user context.It is not an error if the specified user is not currently logged on (and this includes accounts that are not even defined). In this case execution is deemed successful but the method does nothing and returns no data.The specified user must be currently logged on and running *explorer.exe* (Windows Explorer).

script_params ... (optional): Zero or more "anonymous" parameters (i.e. just values, with no " *name:*" prefix) that will be passed directly to the script as its command line parameters. They can be of various types (string, int, etc.). The script parameters can be included anywhere in the list of method parameters, because they are anonymous.

Return values

If parameter *InterpretAsJson* was *true*, the data table produced.

Otherwise:

ExitCode (int): The exit code from execution of the script. -2 if the script timed out.

> **Warning**
>
> Depending on the script, an exit code of 0 does not necessarily mean that execution was successful.

Output (string): The text produced by the script's execution on standard output and standard error streams.

Example

The first PowerShell example runs a script that itself takes 2 positional parameters with values "*script param 1*" and *42*; abandon (but do not kill) the script if it does not end within 5 seconds. Run the script regardless of PowerShell execution policy.

This also shows that usually a call to the *Scripting.Run* method is used in combination with the httpGetFile() function so that the agent itself supplies the local copy of the script file.

```
@resource1 = HttpGetFile(URL:
"myscript.ps1", Size:483,
Hash:"ff3f4bd8fb4e0e0ae0caeef
8eb03cbc6dbf4d61b005bb5f43427
710286811803");
Scripting.Run(Language: "Powe
rShell", LanguageVersion: "3.
0", ScriptPath: @resource1,
TimeoutSecs: 5, ExecutionPoli
cy: "Override", "script param
1", 42);
```

This next PowerShell example runs the script as each of the logged-on users (if any):

AsUser example

```
@script = HttpGetFile(URL:"my
script.ps1", Size:483,
Hash:"ff3f4bd8fb4e0e0ae0caeef
8eb03cbc6dbf4d61b005bb5f43427
710286811803");

@users = Users.GetLoggedOnUse
rs();
```

Module	Scripting

	`@allExecutions = FOREACH @user IN @users DO @account = SELECT Account FROM @user; Scripting.Run(Language: "PowerShell", LanguageVersion: "3.0", ScriptPath:@script, TimeoutSecs:5, AsUser:@account, "script param 1", 42); DONE;`
	
	This Bash example is rather artificial but shows the creation of a temporary bash script then execution of the script to produce "*Hello*" as the *Output* value (and an *ExitCode* of *0*).
	
	`NativeServices.RunCommand(CommandLine: "/bin/bash -c 'echo echo Hello > /tmp/temp'"); Scripting.Run(Language:"Bash", ScriptPath:"/tmp/temp");`

Platforms	• Windows - PowerShell • Linux - bash • MacOS - bash
Notes	This is an alternative to the Scripting.RunText method which instead uses script text supplied as a parameter. Because the 1E Client has multiple workers and so can run a number of instructions in parallel, it is possible that conflicting scripts could run at the same time, e.g. each attempting to install software. (Software packages can only usually be installed one at a time.) If this is likely to be a problem, the 1E Client's "*Exclusivity*"

configuration value should be set to *true* which will ensure that only one script can run at any time across all workers, causing other scripts to be temporarily suspended.By default *Exclusivity* is *false*.This method does not have a SplitLines parameter. If the script output is a single text block and not JSON, then you can use the Utilities.SplitLines method to split the text block into lines.PowerShell script parameters are in effect surrounded by single quotes when the script is executed. This is to allow parameter values to contain embedded spaces, and to allow what can be metacharacters (e.g. '*(*' for PowerShell or '*$*' for bash) to be treated as literals. This means that PowerShell script parameters cannot contain embedded single quotes. bash scripts can contains embedded quotes.If the 1E Client is shut down while a script is running (with or without a timeout specified), the script subprocess is detached from the 1E Client (so that 1E Client shutdown is not blocked), and the Tachyon instruction's status is set to "execution failed" because the 1E Client cannot know the ultimate fate of the script. It is up to the user to decide whether to resubmit the instruction or not when the 1E Client resumes.By default, a long-running instruction will time out. "Long running" is 3600 seconds, i.e. 1 hour. This is to prevent a hanging script from completely blocking the 1E Client, because the 1E Client has an exclusivity feature for safety such

Module	Scripting
	that even if it runs with multiple workers then executing a script prevents any other worker from running an instruction in parallel. (This is because instructions that follow the script's execution might depend on that script having run to completion.) A script hanging indefinitely would completely block the 1E Client, including any instructions to kill the hanging script's process. If a script really must be run to completion, set *TimeoutSecs* to 0 to indicate no timeout.

1.6.1.140 Security.CheckSimpleIoc

Method	**CheckSimpleIoc**
Library	Security
Action	Evaluates an IOC (indicator of compromise) definition on a device.
Parameters	*IpRange* (string; optional): One or more IP ranges, comma delimited without spaces. IPv6 ranges are not currently supported. For example: "*5.5.6.7-5.5.7.2,70.50.72.224-70.50.72.226*" *IpAddress* (string; optional): One or more IP addresses, comma delimited without spaces. IPv6 addresses are not currently supported, but a match will be attempted. For example: "*5.5.6.7,5.5.7.2,70.50.72.224,70.50.72.226*" *Ports* (string; optional): One or more ports, comma delimited without spaces. All ports supplied apply to all *IpRanges* and *IpAddresses* supplied. For example: *"5556,5557,5558,7229,27015"* *Domain* (string; optional): One or more domains, comma delimited without spaces. We don't currently have port information tied to domain lookups. As such, if a device has attempted to resolve a domain in this list, it will be returned as a match. For example: *"domain.com,contoso.malwareland.com,lync.company.discover.com"*. All domains supplied must be lower case.

Method	**CheckSimpleIoc**
	FileSpec (string; optional): One or more *FileSpec* strings, comma delimited without spaces. Such a definition is a filename with extension, followed by a colon and then the file's MD5 hash. For example: *"malerware.exe:5a766a45158c640023d221005082cbb8,actual.txt:5a766a45158c640023d221005082cbb7"*. *FileSpec* matches are done live, and do not use the agent historical data. As such, for this IOC definition, the *DaysAgo* parameter is ignored. All md5 hashes supplied must be lower case. *Url* (string; optional): This parameter is unsupported and will be ignored. *DaysAgo* (integer; optional, default *7*): An integer between 1 and 30 which governs how far back in the agent historical data to look for IOC matches. If a *FileSpec* IOC definition is supplied along with others, this parameter will be ignored only for the *FileSpec* resolution as the FileSpec resolution is always done live.
Return values	FoundIoc (string): Whether or not the device is considered vulnerable given the definition supplied. Possible values are: *"true", "false"*
Example	`Security.CheckSimpleIoc(IpAddress: "5.5.5.10,127.0.0.1,10.10.0.254", Ports: "4000,35525,443", FileSpec: "malerware.exe:5a766a45158c640023d221005082cbb8, actual.txt:5a766a45158c640023d221005082cbb7,", Domain: "notadomain.com,hackerland.cn,google.co.in,lyncdiscoverinternal.1e.com", IpRange: "225.225.226.222-225.225.229.228,127.0.0.0-127.0.0.6");`
Platforms	• Windows
Notes	If multiple parameters are supplied, the definitions are considered "OR"-ed together. If any definition is matched, true will be returned.

1.6.1.141 Security.ModifyQuarantineWhitelist

Method	**ModifyQuarantineWhitelist**
Module	Security
Library	Security
Action	Allows connections from additional specified IP addresses to an already-quarantined device. This allows security tools other than Tachyon to diagnose and protect the device.

Method	**ModifyQuarantineWhitelist**
	It is an error, and the method will fail, if the device is not already in a quarantined state. **Warning** Ensure you understand the impact of using the Quarantine feature. The quarantine methods are not suitable for use from Tachyon Instruction Management Studio (TIMS), mainly because of what they do (cut off the device from most connections) but also because no Tachyon Switch is involved with TIMS. If you must use this method in TIMS (which is not recommended) then you will need to give a *true* value to the *Force* parameter. For more information about using the Quarantine feature, please refer to Tachyon Explorer - Tachyon Quarantine.
Parameters	*AddedIpAddresses* (string): A comma-separated list of IPv4 addresses. Allow connections from these devices as well as those already in the quarantine whitelist.
Return values	*Status* (string): The quarantine status. If the response is a success, this will be *Quarantined*.
Example	Allow connections from device at IP address 10.10.99.150 in addition to those already allowed. `Security.ModifyQuarantineWhitelist(AddedIpAddresses: "10.10.99.150");`
Platforms	• Windows
Notes	

1.6.1.142 Security.DeleteCertificates

Method	**DeleteCertificates**
Module	Security
Library	Security
Action	Deletes every certificate in the local machine cert store (as the agent is run as Local System) with the specified thumbprint.

Method	**DeleteCertificates**
Parameters	*Thumbprint* (string): The SHA1 thumbprint of the certificate. **Note** If the thumbprint string is copied from the Crypto API viewer in Windows, the thumbprint is preceded by a "left to right mark" (Unicode 0x200E / UTF8 surrogates 0xE2 0x80 0x8E - which can look like "â€Ž" in a non-unicode aware editor. In a unicode aware text field this character is one pixel width but invisible on screen unless selected). The agent will handle this character if present, but please be aware of this "invisible" character if you're doing any of your own string comparison in the agent language. *Store* (string; optional): The name of the store certificates are to be deleted from. It is preferred that this is the actual name of the store. If a friendly name is supplied, the agent will attempt to map it to a store name. For example, 'Personal' would be mapped to 'My'. The agent can translate a number of default store names. If the agent cannot translate the name supplied, it will attempt to use the supplied string as the actual name of the store and open it anyway. If this is not supplied, the agent will iterate through predefined system stores (and thus you may actually get multiple hits for a particular certificate if it exists in multiple stores).
Return values	*Subject*(string): The Subject field of the certificate. This contains the Common Name of the certificate, but may also contain other entries. *Thumbprint*(string): This is a SHA1 hash of the certificate content and the certificate serial number. *StoreName*(string): This is the store the certificate was retrieved from. *DeletionTime*(string): This is the time the certificate was marked for deletion. Please note that the certificate is marked as deleted - they not 'retrievable' once deleted. In Microsoft's words, ' *all subsequent attempts to get or find that certificate in that store will fail'* . The certificate will actually be deleted once the context's reference count reaches 0.
Example	`Security.DeleteCertificates(Store: "Intermediate Certification Authorities",` `Thumbprint: "ff01505a76635ede2e1815cc7312e1abde0cddcc");`
Platforms	• Windows
Notes	

1.6.1.143 Security.GetCertificates

Method	**GetCertificates**
Library	Security
Action	Returns every certificate in the local machine cert store (as the agent is run as Local System).
Parameters	*Store* (string; optional): The name of the store certificates are to be retrieved from. The short name of the store is preferable. If you use a long (friendly) name, the agent will attempt to map it to a store name. For example, 'Personal' would be mapped to 'My'. The agent can translate a number of default store names. If the agent cannot translate the name supplied, it will attempt to use the supplied string as the actual name of the store and open it anyway. If this is not supplied, the agent will iterate through predefined system stores (Intermediate Certification Authorities, Trusted Root Certification Authorities, Third-Party Certification Authorities, Enterprise Trust, Other People, Trusted People, Trusted Publisher and Untrusted Certificates), and thus you may actually get multiple hits for a particular certificate if it exists in multiple stores.
Return values	*FriendlyName* (string): The "friendly name" field associated with the certificate. **Note** This is not actually part of the certificate, and a certificate could have a different friendly name associated with it on different devices, as it can be changed after issuing. *Subject* (string): The Subject field of the certificate. This contains the Common Name of the certificate, but may also contain other entries. *Issuer* (string): The Issuer field of the certificate. This contains the Issuer of the certificate. *IssuerThumbprint* (string): The thumbprint (SHA1 hash) of the issuing certificate. Empty if this is a root certificate or the issuer is unavailable. **New in v8.1**. *Thumbprint* (string): This is the SHA1 hash of the certificate, it identifies the certificate and is globally unique. *EffectiveDate* (string): This is the date at which the certificate becomes valid. ('*NotBefore*')

Method	**GetCertificates**

ExpirationDate (string): This is the date at which the certificate is no longer valid. ('*NotAfter*')

SerialNumber (string): This is the serial number of the certificate. Do NOT use this to uniquely identify a certificate.

> **Note**
>
> SerialNumber is only unique to the CA that issued the certificate; it is not globally unique.

StoreName(string): This is the long name of the store the certificate was retrieved from.

Example

Get information about all certificates in the local machine's certificate store:

```
Security.GetCertificates();
```

Platforms

- Windows

Notes

Use the Windows command *certutil -enumstore* to see mappings of the short name to long (friendly) name of all the certificate stores on your PC.

For example:

My "Personal"

Root "Trusted Root Certification Authorities"

Trust "Enterprise Trust"

CA "Intermediate Certification Authorities"

TrustedPublisher "Trusted Publishers"

Disallowed "Untrusted Certificates"

AuthRoot "Third-Party Root Certification Authorities"

TrustedPeople "Trusted People"

ClientAuthIssuer "Client Authentication Issuers"

ADDRESSBOOK "Other People"

Remote Desktop "Remote Desktop"

REQUEST "Certificate Enrolment Requests"

Method	**GetCertificates**
	SmartCardRoot "Smart Card Trusted Roots" *TrustedAppRoot "Trusted Packaged App Installation Authorities"* *TrustedDevices "Trusted Devices"*

1.6.1.144 Security.GetQuarantineStatus

Method	**GetQuarantineStatus**
Library	Security
Action	Gets the current quarantine status of the device. **Warning** Ensure you understand the impact of using the Quarantine feature. The quarantine methods are not suitable for use from Tachyon Instruction Management Studio (TIMS), mainly because of what they do (cut off the device from most connections) but also because no Tachyon Switch is involved with TIMS. If you must use this method in TIMS (which is not recommended) then you will need to give a *true* value to the *Force* parameter. For more information about using the Quarantine feature, please refer to Tachyon Explorer - Tachyon Quarantine.
Parameters	
Return values	*Status* (string): The quarantine status. This can be one of four values: • *Quarantined*: The device is currently under quarantine • *NotQuarantined*: The device is not currently under quarantine • *UnknownValue*: The device may be under quarantine, but persistent storage may be damaged, as some values are unexpected • *UnknownNoSPS*: The device is unable to read its persistent storage. Quarantine status is unknown, and the device cannot be quarantined (or unquarantined) until persistent storage is available.
Example	`Security.GetQuarantineStatus();`
Platforms	• Windows

Method	**GetQuarantineStatus**
Notes	• Quarantine requires working persistent storage. • If persistent storage is removed or corrupted during quarantine, the agent will be unable to revert to an unmodified state • Upgrading the agent whilst under quarantine is not supported or recommended.

1.6.1.145 Security.QuarantineDevice

Method	**QuarantineDevice**
Module	Security
Library	Security
Action	Attempts to force the device to only communicate using Tachyon. **Warning** Ensure you understand the impact of using the Quarantine feature. The quarantine methods are not suitable for use from Tachyon Instruction Management Studio (TIMS), mainly because of what they do (cut off the device from most connections) but also because no Tachyon Switch is involved with TIMS. If you must use this method in TIMS (which is not recommended) then you will need to give a *true* value to the *Force* parameter. For more information about using the Quarantine feature, please refer to Tachyon Explorer - Tachyon Quarantine. • Routing tables will be modified. These can be reset by disabling and re-enabling the adapter, or by restarting the machine. Doing so will break quarantine until the next quarantine enforcement period. • The hosts file will be modified whilst the device is in quarantine. • Agent communications will be limited to IPv4, and only communication with the Switch servers, Background Channel servers, and any designated additional IP addresses will be allowed. • IPv6 will be disabled for all adapters.

Method	**QuarantineDevice**

- Changes made to routing tables, the hosts file, or IPv6 bindings during quarantine will be lost, as the agent will attempt to revert modifications it makes to a pre-quarantine state.

Parameters

AddedIpAddresses (string; optional, default empty): A comma-separated list of IPv4 addresses. For remediation when quarantined, allow connections from these devices as well as the Tachyon Switch and Background Channel host(s).

Force (boolean; optional, default *false*): Once a device has been quarantined, only Tachyon can take it out of quarantine because the device is isolated from everything else. This requires that at least 1 Switch and at least 1 Background Channel must be resolvable to an IP address, otherwise a quarantined device would become completely cut off. By default if this condition is not satisfied then the device will not be quarantined because it cannot be restored. However, setting *Force* to *true* allows a device to be quarantined regardless, with the risk that it cannot then be unquarantined.

Warning

Using *Force* risks making un-quarantining using Tachyon impossible.

Return values

Status (string): The quarantine status. If the response is a success, this will be *Quarantined*.

Note

If the device was already in a quarantined state, the action is successful but nothing is returned ("success no content").

Example

```
Security.QuarantineDevice();
```

Platforms

- Windows

Notes

- Usually at least one Switch host and at least one Background Channel host *must* be resolvable to an IP address (so that Tachyon will be able to un-quarantine the device), otherwise the method fails. (New in v5.1.) This can be overridden with the *Force* parameter.

Warning

The Background Channel URL (including the port number) is not checked, so it is possible that a subsequent security remediation instruction that uses it will not work. However, a simple instruction to just unquarantine the device will work fine because it would not need the Background Channel.

Method	**QuarantineDevice**

- Quarantine requires the Agent's persistent storage.
  - If persistent storage is removed or corrupted during quarantine, the agent will be unable to revert to an unmodified state.
- Communication with the Agent can only occur over IPv4.
  - If an IPv4 address cannot be resolved for at least one Switch and at least one Background Channel URL, quarantine will not be enforced.
- The quarantine enforcement interval while the device is not under quarantine can be modified in Agent configuration by setting *`Module.Security.QuarantineEnforcementIntervalSeconds`*.
- Upgrading the Agent whilst under quarantine is not supported and may cause quarantine to be permanent!

Warning

CRL checks must be set to soft to use the quarantine feature - a CRL expiry can cause the device to become uncontactable. If certificate expiry occurs under Quarantine, the device may become uncontactable. Quarantine will still be in effect in both cases, however.

For more information about using the Quarantine feature please refer to Tachyon Quarantine.

1.6.1.146 Security.UnquarantineDevice

Method	**UnquarantineDevice**
Module	Security
Library	Security
Action	Allows devices to communicate with devices other than the Tachyon Switch and Background channel servers. This is intended to be used after QuarantineDevice. Changes made to routing tables, the hosts file, or IPv6 bindings during quarantine will be lost, as the agent will attempt to revert modifications it made back to a pre-quarantine state. **Warning**

Method	**UnquarantineDevice**
	Ensure you understand the impact of using the Quarantine feature. The quarantine methods are not suitable for use from Tachyon Instruction Management Studio (TIMS), mainly because of what they do (cut off the device from most connections) but also because no Tachyon Switch is involved with TIMS. If you must use this method in TIMS (which is not recommended) then you will need to give a *true* value to the *Force* parameter. For more information about using the Quarantine feature, please refer to Tachyon Explorer - Tachyon Quarantine.
Parameters	*Reboot* (bool, optional, default *false*): Reboot the device after Unquarantine.
Return values	*Status* (string): The quarantine status. If the response is a success, this will be *NotQuarantined*. **Note** If the device was already in a quarantined state, the action is successful but nothing is returned ("success no content").
Example	`Security.UnquarantineDevice();` `Security.UnquarantineDevice(Reboot: true);`
Platforms	• Windows
Notes	• Quarantine requires working persistent storage. – If persistent storage is removed or corrupted during quarantine, the agent will be unable to revert to an unmodified state. • Upgrading the agent whilst under quarantine is not supported or recommended. • Rebooting device after the Unquarantine operation can solve localhost connection issues

1.6.1.147 Shopping.LaunchWsaWizard

Method	**LaunchWsaWizard**
Module	Shopping
Library	Shopping

Method	**LaunchWsaWizard**
Action	Launch the Shopping WSA wizard.
Parameters	*RefId* (string, optional): An id to uniquely identify a WSA order in a third-party system. E.g it could be a ticket id in SerivceNow. This id is saved in the Shopping database once an order is created. If omitted, the id saved in the Shopping database will be null. *User* (string, optional): The user in whose session WSA wizard should be launched. It should be in the domain\username format. If omitted, WSA wizard will be launched in the session of the user with an active session.
Return values	The returned result will have this form: *Launched* (bool): True if WSA wizard is launched successfully, false otherwise. *Status* (string): "Successfully launched" if wizard is launched successfully, an error description otherwise.
Example	`Shopping.LaunchWsaWizard(RefId:"RITM00001", User:"ACME\John");` where RITM00001 is a ticket id in a third party system like ServiceNow. John is a user in ACME domain.
Platforms	• Windows
Notes	**Warning** This method is reserved for use by 1E. It requires the Shopping module to be enabled in 1E Client. It also requires a full Shopping infrastructure to be installed.

1.6.1.148 Shopping.ShowNotification

Method	**ShowNotification**
Module	Shopping
Library	Shopping
Action	Notify the Shopping user with details of Order Status using Shopping notification icon and/or toast popup.
Parameters	*title* (string, required): The title of the notification to be shown to the user.

description (string, required): The description of the notification to be shown to the user.

userName (string, optional): The user in whose session the notification will be shown. It should be in the domain\username format. If omitted, notification will be shown to all the users with an active session.

showInSystemTray (string, required): If "true", the notification will be shown in the Shopping UI's notifications panel.

showToastNotification (string, required): If "true", the notification will be shown as a toast notification.

isIntermediate (string, required): If "true", this notification will be treated as an intermediate notification. An intermediate notification is not shown immediately, but after a certain time has passed. If a new ShowNotification is received while an intermediate notification is in its waiting period, the old notification will never be shown. The waiting period is configurable in the setting
After how many minutes would you like the next notification to show on the Manage Order Tracking page in Shopping.

Method	**ShowNotification**
isLast (string, required): If "true", this notification will be treated as the last notification. For a last notification, no delay notification is shown. *showDelayNotification* (string, required): If "true", a delay notification will be shown if a certain time has passed after this notification was shown and no new notification was received during the waiting period. The waiting period is configurable in the setting *After how many minutes would you like the next notification to show* on the Manage Order Tracking page in Shopping. *orderId* (string, required): The order id of the order for which this notification is shown.	
Return values	The returned result will have this form: *SessionId* (integer): An integer representing the session id of the user to whom the notification was shown. *UserName* (string): The name of the user to whom the notification was shown.
Example	`Shopping.ShowNotification( Title: "%title%", Description: "%description%", UserName: "%userName%", ShowInSystemTray: "%showInSystemTray%", ShowToastNotification: "%showToastNotification%`

Method	**ShowNotification**
	", IsIntermediate: "%isIntermediate%", IsLast: "%isLast%", ShowDelayNotification: "%showDelayNotification%", OrderId: "%orderId%")
Platforms	• Windows
Notes	**Warning** This method is reserved for use by 1E. It requires the Shopping module to be enabled in the 1E Client. It also requires a full Shopping infrastructure to be installed.

1.6.1.149 Software.GetInstallations

Method	**GetInstallations**
Module	Software
Library	Core
Action	Gets a list of the installed software available to all users, optionally filtered by product and/or publisher.
Parameters	*Product* (string; optional): If specified, search for just this product.

Method	**GetInstallations**
	Publisher (string; optional): If specified, get just this publisher's software.
Return values	A table of: • *Product* (string): The product name. • *Publisher* (string): The publisher of the product. • *Version* (string): The version of the product. • *InstallDate* (string): If available, when the product was installed. • *Architecture* (string): Platform architecture of this installation (e.g. x86, x64)

Example	Get all software:
	`Software.GetInstallations();`
	Find and remove all installed instances of Calculator Plus:
	`Software.GetInstallations( Publisher : "Microsoft", Product : "Microsoft Calculator Plus");`
	`evaluate;`
	`Agent.Echo( Message: "Is installed");`
	`Software.Uninstall( Publisher : "Microsoft", Product : "Microsoft Calculator Plus");`
	To select a specific version, use a SQL WHERE clause:
	`@installs = Software.GetInstallations( Publisher : "Microsoft", Product : "Office Professional");`
	`SELECT DISTINCT * FROM @installs WHERE Version LIKE "16.0%";`

Platforms	• Windows • Linux • MacOS
Notes	To find additional software installed for specific users, use Software.GetUserInstallations.

1.6.1.150 Software.GetUserInstallations

Method	**GetUserInstallations**
Library	Core
Action	Gets a list of the software installed for specific users, optionally filtered by product and/or publisher.
Parameters	*Product* (string; optional): If specified, search for just this product. *Publisher* (string; optional): If specified, get just this publisher's software.
Return values	A table of: • *Architecture* (string): Platform architecture of this installation (e.g. x86, x64) • *InstallDate* (string): If available, when the product was installed. • *Product* (string): The product name. • *Publisher* (string): The publisher of the product. • *Sid* (string): The user identifier for whom this software has been installed. • *Username* (string): Friendly representation of the *Sid* - this will only be resolved locally. • *Version* (string): The version of the product.
Example	Get all user specific software: `Software.GetUserInstallations();`
Platforms	• Windows
Notes	• *Sid* resolution may fail as this method will not go over the wire. In this case Username will be NULL. • The file modified and accessed time attributes of the NTUSER.dat directory entries under the %USERPROFILE% folder for each user and service principal might be updated to some time during the execution of this method. This is a side effect of loading and unloading the user hive and is housekeeping performed by Windows, not the 1E Client. This can cause "unused" profiles to appear to be used if the modified times are employed

Method	**GetUserInstallations**

to determine whether a user has logged in, which is not a reliable method of establishing this fact.

- This method reads user uninstall registry keys from HKU, but will not load hives.
- Installs for this function are retrieved from *HKU:\<SID>\Software\Microsoft\Windows\CurrentVersion\Uninstall*

1.6.1.151 Software.Uninstall

Module	Software
Library	Core
Action	Uninstall the specified software item.
Parameters	*Product* (string): The software product to uninstall. Any instances that match will be uninstalled. *Publisher* (string): The publisher of the product to uninstall. This value must be empty if there is no recorded publisher for the product. *Version* (string; optional): If given, uninstall only this exact version of the product. Leading zeroes are not significant in dot-separated numeric fields, so for example versions "*11.0.0*" and "*11.0.00*" are considered to be equivalent. However, there must be the same quantity of dot-separated fields, e.g. "*11*" and "*11.0*" are *not* equivalent. Version is not relevant for uninstall on some platforms, e.g. those with Debian-style packages.
Return values	*CommandLine* (string): The command(s) used to uninstall the software.

Example	Remove all installed instances and versions of Calculator Plus:
	
	`Software.GetInstallations( Publisher : "Microsoft", Product : "Microsoft Calculator Plus");`
	`evaluate;`
	`Software.Uninstall( Publisher : "Microsoft", Product : "Microsoft Calculator Plus");`

Platforms	• Windows

Module	Software
	• Linux
Notes	On Windows, only uninstall commands that use *msiexec* are supported.

1.6.1.152 Storage.Check

Method	**Check**
Module	Storage
Library	Core
Action	Tests the existence of a user defined persistent storage table.
Parameters	*Name* (string): The name of the persistent storage table. Case is not significant.
Return values	Name (string): The table's name if the storage table exists in persistent storage, else nothing.

Example	`@storageName = SELECT "LoggedInAudit" AS Name;`
	`@found = Storage.Check(Name:@storageName);`
	`IF (@found)`
	`  @currentEntries = Storage.Get(Name:@storageName);`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	This method can be used with the SCALE *evaluate* statement to abandon processing if the persistent storage table is not present. See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.153 Storage.List

Method	**List**
Module	Storage
Library	Core
Action	Enumerates all user-defined persistent storage tables.
Parameters	(None)
Return values	Name (string): The name of each table.

Example	`// Remove all user-defined storage values with the given temporary prefix`
	`@names = Storage.List();`
	`@names = SELECT * FROM @names WHERE Name LIKE "MyTempData.%";`
	
	`@deleted = FOREACH @name IN @names`
	`DO`
	`  Storage.Delete(Name: @name);`
	`DONE;`
	
	`SELECT "Deleted entry " || Name AS Result FROM @deleted;`

Platforms	• Windows • Linux • MacOS
Notes	The names returned by this method may not match the case of their original names because user-defined persistent storage table names are handled with no case-sensitivity. See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.154 Storage.Delete

Method	**Delete**
Module	Storage
Library	Core
Action	Removes an existing user defined persistent storage table.
Parameters	*Name* (string): The name of the persistent storage table. Case is not significant.
Return values	Name (string): The table's name if the storage table is deleted from persistent storage, else nothing.

Example	`@storageName = SELECT "LoggedInAudit" AS Name;`
	
	`@result = Storage.Delete(Name:@storageName);`
	`IF (@result)`
	`    SELECT @storageName.Name ||" Table deleted" AS Status FROM @st`
	`orageName;`
	`ELSE`
	`    SELECT @storageName.Name ||" Table not found" AS Status FROM @`
	`storageName;`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	This method can be used with the SCALE *evaluate* statement to abandon processing if the persistent storage table is not present. See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.155 Storage.Get

Method	**Get**
Module	Storage
Library	Core
Action	Indicate whether a persistent storage table of the specified name is present and return its contents if present.
Parameters	*Name* (string): The name of the persistent storage table. Case is not significant.
Return values	*Value* (table): Contents of the persistent storage table, which may be empty, either because the table is empty in persistent storage or because the table does not exist.

Example	`@found = Storage.Check(Name: "2018.Sept.AdobeUsers");`
	`IF NOT (@found)`
	`  SELECT "Table not found" AS Status;`
	`ELSE`
	`  @table = Storage.Get(Name: "2018.Sept.AdobeUsers");`
	`  IF NOT (@table)`
	`  SELECT "Table is empty" AS Status;`
	`  ELSE`
	`  SELECT "Table retrieved and has content" AS Status;`
	`  ENDIF;`
	`ENDIF;`

Platforms	• Windows • Linux • MacOS
Notes	This method can be used with the SCALE *evaluate* statement to abandon processing if the persistent storage table is not present. See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.156 Storage.GetRemote

Method	**GetRemote**
Module	Storage
Library	Core
Action	Retrieve the requested datum directly from the Platform central repository.
Parameters	*Name* (string): The key of the datum to retrieve. Case is significant. *TimeoutSecs* (integer; optional, default 60): If the operation has not completed within the specified time period, it will fail with an error.
Return values	*Value* (table): Contents of the value retrieved as a string, which may be empty, either because the value is empty or because the key does not exist.
Example	`@token = Storage.GetRemote(Name: "Api.Token");`
Platforms	• Windows • Linux • MacOS
Notes	It is not an error if the datum named by the key does not exist, but an empty result is returned. This method **does not return a value in TIMS** because it requires a Tachyon Switch. See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.157 Storage.Set

Method	**Set**
Module	Storage
Library	Core
Action	Set or change the value of the named persistent storage table.
Parameters	*Name* (string): The name of the persistent storage table. Case is not significant.

Value (table): The table contents to be written to, or overwritten in, the named persistent storage table.

Append (bool; optional, default *false*): Whether to append to whatever is already in the table (true) or overwrite it (false).

It is not an error to append if the table does not already exist; in this situation the parameter's value is irrelevent.

> **Warning**
>
> There is no check that the schema of the appended rows is the same as that of those already there.

MaxRowCount (int; optional, default *1000*): When appending, limit the table to this maximum number of rows.

The range is 1 to the SelectRowsLimit configuration value (100000 by default). This can be specified only if *Append* is true. There is no limit when overwriting (i.e. when *Append* is explicitly or implicitly *false*).

> **Note**
>
> Regarding truncation behaviour, the new (appended) data is considered more important than the old (original)

Method	**Set**
data, so rows are deleted from the *front* of the table. Also, after the append there will be no more than *MaxRowCount* rows left in the table. For example, if the table originally contained rows 1 to 10 then we append rows 11 to 15 with *MaxRowCount* set to 6 then we end up with a table containing rows 10 to 15.	
Return values	*Name* (string): The name of the new or updated persistent storage table.

Examples	A simple example.
	`@table = Agent.GetSummary();`
	`Storage.Set(Name: "Agent_Summary.Oct2018", Value: @table);`
	The method overwrites whatever was previously in the table. If you want to **accumulate** data by appending, these show examples of a task that would be run periodically to record who is logged on at that time. They also return the accumulated data.
	Using Append
	`@loggedOnUsers = Users.GetLoggedOnUsers();`
	`@newEntries = SELECT *, datetime("now") AS Timestamp FROM @loggedOnUsers;`
	`@storageName = SELECT "LoggedInAudit" AS Name;`
	`// Store data`
	`Storage.Set(Name:@storageName, Value:@newEntries, Append:true);`
	`Storage.Get(Name:@storageName);`

Platforms	• Windows • Linux

Method	**Set**
	• MacOS
Notes	See User Defined Persistent Storage page for guidance using persistent storage tables.The following methods existing in the Agent Storage module: Storage.Check — Tests the existence of a user defined persistent storage table. Storage.Delete — Removes an existing user defined persistent storage table. Storage.Get — Indicate whether a persistent storage table of the specified name is present and return its contents if present. Storage.GetRemote — Retrieve the requested datum directly from the Platform central repository. Storage.List — Enumerates all user-defined persistent storage tables. Storage.Set — Set or change the value of the named persistent storage table.

1.6.1.158 Tagging.Check

Module	Tagging
Library	Core
Action	Tests the existence of a named tag of a specified type, optionally with the specified value.
Parameters	*Name* (string): The name of the tag to investigate. *Value* (string; optional): The value of the tag if we want to test whether the tag has a particular value. *Scopable* (boolean; optional): • *true* - Check a Device tag (default) • *false* - Check a Freeform tag
Return values	*Value* (string): The tag's value if the tag exists (and has the specified value), else nothing.

Module	Tagging

Example	`// Get the value of the freeform tag named "WithScanner", if it exists`
	`Tagging.Check(Name: "WithScanner", Scopable: false);`
	
	`// Find instances of freeform tag "WithScanner" that are set to "WorkForce DS1630":`
	`Tagging.Check(Name: "WithScanner", Value: "WorkForce DS1630", Scopable: false);`

Platforms	• Windows • Linux • MacOS
Notes	This method can be used with the agent language's *evaluate* statement to abandon processing if the tag is not present, or does not have the specified value. See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.159 Tagging.Clear

Module	Tagging
Library	Core
Action	Delete all tags of the specified type.
Parameters	*Scopable* (boolean; optional): • *true* - Delete all Device tags (default) • *false* - Delete all Freeform tags

Module	Tagging
Return values	*Success* (boolean): Whether deletion of all tags of the type was completely successful or not.
Example	`// Delete all freeform tags` `Tagging.Clear(Scopable: false);`
Platforms	• Windows • Linux • MacOS
Notes	**Note** **Use with care!** This method is recommended only for testing. See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.160 Tagging.Count

Module	Tagging
Library	Core
Action	Get the quantity of tags of the specified type.
Parameters	*Scopable* (boolean; optional): • *true* - Count number of Device tags (default) • *false* - Count number of Freeform tags
Return values	*Count* (int): The number of tags of that type.

Module	Tagging
Example	`// Return the number of freeform tags` `Tagging.Count(Scopable: false);`
Platforms	• Windows • Linux • MacOS
Notes	See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.161 Tagging.Delete

Module	Tagging
Library	Core
Action	Delete the named tag of the specified type, and indicate if it originally existed.
Parameters	*Name* (string): The name of the tag to delete. *Scopable* (boolean; optional): • *true* - Delete a Device tag (default) • *false* - Delete a Freeform tag
Return values	*Name* (string): The tag's name. *Present* (boolean): *true* if the tag originally existed (and so was deleted), *false* if there was no such tag.
Example	`// Delete the "WithScanner" freeform tag` `Tagging.Delete(Name: "WithScanner", Scopable: false);`

Module	Tagging
Platforms	• Windows • Linux • MacOS
Notes	It is not an error if the tag does not exist. See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.162 Tagging.Get

Method	**Get**
Module	Tagging
Library	Core
Action	Indicate whether a named tag of the specified type is present and return its value if present.
Parameters	*Name* (string): The name of the tag whose value is to be retrieved. *Scopable* (boolean; optional): • *true* - Get a Device tag (default) • *false* - Get a Freeform tag
Return values	*Present* (boolean): *true* if the tag originally exists and so has a value, *false* if there was no such tag. *Value* (string): The tag's value if it was present, otherwise an empty string. An empty string is also a valid value.

Method	**Get**
Example	`// Get the value of the "WithScanner" freeform tag` `Tagging.Get(Name: "WithScanner", Scopable: false);`
Platforms	• Windows • Linux • MacOS
Notes	It is not an error if the tag does not exist. See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.163 Tagging.GetAll

Module	Tagging
Library	Core
Action	Fetch the tag list (containing all tags) of the specified type.
Parameters	*Scopable* (boolean; optional): • *true* - Get the list of Device tags (default) • *false* - Get the list of a Freeform tags
Return values	*Values* (string): A single string with all tags formatted as in an online message, ordered alphabetically by name. The format is "*|name1=value1|name2=value2|...|namen=valuen|*", i.e. name-value pairs are separated by the '=' character, and vertical bars separate each name-value pair and also start and end the list. The "alphabetical" order is not rigidly defined. An empty list (i.e. when there are no tags of that type at all) results in "||".

Module	Tagging
	Note Tag names are stored internally in upper case, and so they are also listed in that form.
Count (int): The number of key-value pairs that exist for the specified tag type. This will return the same information as Tagging.Count(). v3.0.100+	
Length(int): The length of the tag string as stored by the agent. v3.0.100+	

Example	`// Get all freeform tags`
	`Tagging.GetAll(Scopable: false);`
	
	`// Get all device tags, using default Scopable:true`
	`Tagging.GetAll();`

Platforms	• Windows • Linux • MacOS
Notes	See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.164 Tagging.Set

Module	Tagging
Library	Core
Action	Set (or change if already set) the named tag of the specified type to the specified value.
Parameters	*Name* (string): The name of the tag to set. Only alphanumeric characters are allowed. *Value* (string): The value to assign to the tag; can be empty. Only alphanumeric characters are allowed (see notes). *Scopable* (boolean; optional): • *true* - Set a Device tag (default) • *false* - Set a Freeform tag
Return values	*Name* (string): The tag's name. *Value* (string): The tag's (new) value.
Example	`// Set the "WithScanner" freeform tag to "WorkForce DS1630"` `Tagging.Set(Name: "WithScanner", Value: "WorkForce DS1630", Scopable: false);`
Platforms	• Windows • Linux • MacOS
Notes	Set() does not allow a NAME=value pair to be set that increases the total size of the scopable tag list above 511 characters, and will return a runtime error. See Tags page for guidance using tags.Device tags are also known as Coverage or Scopable tags. Software tags are not managed using Tagging methods.The following methods existing in the Agent Tagging module: Tagging.Check — Tests the existence of a named tag of a specified type, optionally with the specified value. Tagging.Clear — Delete all tags of the specified type. Tagging.Count — Get the quantity of tags of the specified type. Tagging.Delete — Delete the named tag of the specified type, and indicate if it originally existed. Tagging.Get — Indicate whether a named tag of the specified type is present and return its value if present. Tagging.GetAll — Fetch the tag list (containing all tags) of the specified

Module	Tagging
	type. Tagging.Set — Set (or change if already set) the named tag of the specified type to the specified value.

1.6.1.165 Users.FindLocalGroupMember

Method	**FindLocalGroupMember**
Module	Users
Library	Core
Action	List the local groups of which the specified user is an immediate member.
Parameters	*AccountName* (string): The user account to search for. On Windows, this should be in the format: *domain**user*.
Return values	For each local group that has the account as a member: • – *Account* (string): Matched account name. – *GroupName* (string): The name of a local group that includes the account.
Example	Show which local groups "megacorp\herbert.west" is a member of: `Users.FindLocalGroupMember(AccountName: "megacorp\\herbert.west");`
Platforms	• Windows
Notes	"Immediate member" means that the user is defined explicitly and directly as a member of that group. If there is a group hierarchy involved (e.g. the user is a member of a domain group which is itself a member of a local group), then neither group will appear in the results.

1.6.1.166 Users.GetInboundConnectionHistory

Module	Users
Library	Core
Action	List the historical inbound Terminal Services (RDP) connection history

Module	Users
Parameters	(None)
Return values	For each session: • – *ClientIpAddress* (string): The IP Address of the client machine making the connection – *LastSeen* (datetime): The date and time of the most recent record – *User* (string): The authenticated user account of the connection to this 'server' machine
Example	`Users.GetInboundConnectionHistory();`
Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.167 Users.GetLocalGroupMembers

Method	**GetLocalGroupMembers**
Module	Users
Library	Core
Action	List the members of a local group.
Parameters	*GroupName* (string): The local group to enumerate.
Return values	For each member of the local group: • *Account* (string): Member name. • *AccountType* (string): one of the following: – *User*

Method	**GetLocalGroupMembers**
	– *Group* – *Built-in group* – *Delete* – *Unknown*
Example	List the members of the local Administrators group: `Users.GetLocalGroupMembers(GroupName: "Administrators");`
Platforms	• Windows
Notes	Group names are language specific. For example, English Administrators is not the same as Spanish Administradores.

1.6.1.168 Users.GetLoggedOnUsers

Module	Users
Library	Core
Action	List the active (currently logged-on) user sessions.
Parameters	(None)
Return values	For each session: • *Account* (string): User name. • *SessionId* (string): Session ID. • *SessionName* (string): Session Name. • *LogonTime* (string): Logon time of the session. Only available for Windows Vista SP1 and later. • *SessionState* (string): The session state as defined by the WTS_CONNECTSTATE_CLASS enum. MS Windows hosts only. • *RemoteSession* (boolean): Whether or not the session is a remote session • *RemoteIp* (string): IP address of the client if the connection is remote (otherwise empty).

Module	Users
	• *RemoteName* (string): Hostname of the client if the connection is remote (otherwise empty). On MacOS only the IP address is available. • *UserPrincipal* (string): User principal. Introduced in v24.11 for Linux, MacOS and windows.

Example	`@users = Users.GetLoggedOnUsers();`
	`//get the home drive/path for each logged on user by running a command as each of them`
	`@home =`
	`  FOREACH @user IN @users DO`
	`  NativeServices.RunCommand(CommandLine:"CMD /c ECHO %HOMEDRIVE%%HOMEPATH%",AsUser:@user.Account);`
	`  SELECT usr.*, cmd.Output AS Home FROM @user usr, @@last cmd;`
	`  DONE;`

Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.169 Users.GetOutboundConnectionHistory

Method	**GetOutboundConnectionHistory**
Library	Core
Action	List the historical outbound Terminal Services (RDP) connection history
Parameters	(None)
Return values	For each session: • *ServerIpAddress* (string): The IP Address of the server machine that accepted the connection at the time. • *ServerFqdn* (string): The fqdn of the server machine at the time if it was used instead of an IP Address • *LastSeen* (dateTime): The date and time of the most recent record of this activity

Method	**GetOutboundConnectionHistory**
Example	`Users.GetOutboundConnectionHistory();`
Platforms	• Windows
Notes	

1.6.1.170 Users.GetPrimaryUser

Module	Users
Library	Core
Action	Get the primary user of a device.
Parameters	(None)
Return values	*`PrimaryUser`* (string): The name of the primary user of a device. • Windows: If historic data is available (requires Inventory module to be enabled) - the user with the longest total logged-on duration over the last 28 days. From v23.7 onwards this is the user with the highest total InteractionMinutes in the $DeviceInteraction_Daily_Raw table, and if gives no result then falling back to the user with the highest total Duration in the $UserUsage_Daily table. Prior to v23.7 only the $UserUsage_Daily table was used. Without historic data - whoever has been logged-in longest of all the active users. • macOS and Linux : If historic data is available - the user with the longest total logged-on duration over the last 28 days. Without historic data - the user with the most number of active sessions. *Active* means currently logged on.
Example	`Users.GetPrimaryUser();`
Platforms	• Windows • Linux • MacOS

Module	Users
Notes	The following is ignored: • Service accounts on Windows • Daemon usage on non-Windows Root logins are not ignored. More detail about active (currently logged-on) users is available using Users.GetLoggedOnUsers. For Windows, historic data is also stored in the Tachyon Activity Record: User usage daily table if the Inventory module is enabled. The 28-day threshold used to analyze historical data is not configurable in this version of Tachyon.

1.6.1.171 Utilities.GenerateSequence

Module	Utilities
Library	Core
Action	Build a sequence of numeric values, starting at 1 up to and including the value of a specified limit.
Parameters	*Limit* (integer): The upper bound of the sequence to generate. The range is 0 to 65535 inclusive. If 0 is specified then the resulting sequence will be empty.
Return values	For each numeric value between 1 and Limit inclusive: • *Sequence* (integer): The value in the sequence.

Example	Builds the numbers 1 to 10 inclusive, and then echoes back a message with the corresponding number into the results
	`FOREACH @i IN Utilities.GenerateSequence(Limit: 10)`
	`DO`
	`  @msg = SELECT "This is number " || @i.Sequence AS Message FROM @i;`
	`  Agent.Echo(Message: @msg.Message);`
	`DONE;`

Platforms	• Windows

Module	Utilities
	• Linux • MacOS
Notes	As per the example above, the GenerateSequence method can be useful to achieve FOR-loop-style behaviour using the FOREACH construct.

1.6.1.172 Utilities.JsonFromTable

Module	Utilities
Library	Core
Action	Serializes a table variable into a JSON string
Parameters	*Table* (table): The table variable to serialise.
Return values	*Json* (string): The Json serialisation that represents the table variable supplied.

Example	Serialise a table variable as Json ...
	`@scalar = SELECT 'abc' AS col1;`
	`@scalarJson = Utilities.JsonFromTable(Table:@scalar);`
	`//@scalarJson = '[{"col1":"abc"}]'`
	
	`@tableJson = SELECT 'abc' AS col1 , 'def' AS col2`
	`UNION`
	`SELECT 'ghi' AS col3, 'jkl' AS col4;`
	`@json = Utilities.JsonFromTable(Table:@tableJson);`
	`//@tableJson = '[{"col1":"abc","col2":"def"},{"col1":"ghi","col2":"jkl"}]'`

Platforms	• Windows • Linux • MacOS
Notes	

1.6.1.173 Utilities.SplitLines

Module	Utilities
Library	Core
Action	Convert a delimited fragment of text into multiple rows.
Parameters	*Text* (string): The text value to be converted into multiple rows. *Delimiter* (string; optional, default: *new line*): The value present in *Text* which should be treated as the separator between rows. If *Delimiter* is not passed, or is passed as empty, then the delimiter is assumed to be a new-line sequence. *RemoveEmptyEntries* (boolean; optional, default: false): If set to true, discards any rows which are empty.
Return values	For each occurrence of *Delimiter* within *Text*: • *Output* (string): The text that occurs between the previous delimiter (or the start of the text) and the next delimiter (or the end of the text)

Example	Take a string in CSV format and split that string into multiple rows based on a comma delimiter:
	
	`@c = SELECT "Red,Yellow,Blue,Black,White,Green" AS Colors;`
	`@c = Utilities.SplitLines(Text: @c.Colors, Delimiter: ",");`

Platforms	• Windows • Linux • MacOS
Notes	The *Delimiter* parameter does not have to be a single character; it can be any sequence of text. For example, you could use a delimiter of " " to help split HTML list items into individual rows. If you omit the *Delimiter* parameter to split the text based on new-lines, the Agent will interpret recognized combinations of carriage-return and line-feed characters as a new-line. Line numbers are not returned, therefore you must use the built-in SQLite autoincremented key **rowid** if you need to perform operations on the Output. For example the below code will return the last inserted row "Green" whereas max(Output) would return "Yellow".

Module	Utilities

```
select Output from @c order by rowid desc limit 1;
```

1.6.1.174 Utilities.TableFromJson

Module	Utilities

Library	Core
Action	Deserializes a correctly escaped JSON string that represents a two dimensional set of data with consistent columns and rows into a table variable.
Parameters	*Json* (string): The JSON string that will be converted into a table.
Return values	A table built from the input JSON string, with one column for each JSON property.
Example	Convert some Json that contains two rows and six columns

```
// returns table with 6 columns and 2 rows
@json = "[
{\"id\":1,
 \"first_name\":\"Eugenio\",
 \"last_name\":\"Verbrugghen\",
 \"email\":\"everbrugghen0@mashable.com\",
 \"gender\":\"Male\",
 \"ip_address\":\"188.126.39.115\"},
{\"id\":2,
 \"first_name\":\"Craggy\",
 \"last_name\":\"MoneyPenny\",
 \"email\":\"cmoneypenny@qq.com\",
 \"gender\":\"Male\",
 \"ip_address\":\"142.195.163.141\"}
]";
Utilities.TableFromJson(Json:@json);
```

Platforms	• Windows • Linux • MacOS
Notes	The JSON string must adhere to RFC 8259.

Module	Utilities
	This method will only deserialise a two dimensional table. Any property containing an array or another object instead of a boolean, string, number or null will be discarded. Any property missing from a JSON object but exists in another object, will return a null.

1.6.2 Tachyon Agent Built-in Functions

Alphabetic listing of Agent built-in functions. Under this page are separate pages with full details for each function.

The Agent language includes some cross-platform built-in functions – think of them like methods available in a "global" module. The names of built-in functions are **case-sensitive**.

See the Agent built-in functions version table on the Functions and Methods Reference page to know the earliest version of the Agent in which a built-in function was implemented.

Agent Built-in Functions

- HttpGetFile— A built-in function for downloading content (resources) over HTTP(S) via the Tachyon Background channel, or from an external web-server.
- IsUndefined— Given a table as a parameter, this built-in function will output a table that indicates whether the input table is real and full of values or whether it is the result of an operation that produced nothing, which is different to producing an empty table.
- Raise— A built-in function to raise one or more events.

1.6.2.1 HttpGetFile

Function	**HttpGetFile**
Action	A built-in function for downloading content (resources) over HTTP(S) via the Tachyon Background channel, or from an external web-server. If the file cannot be downloaded, or the downloaded content does not match the hash and size supplied, an error occurs and the instruction is terminated. The file is downloaded into a randomly named sub-directory of TemporaryDirectory. From version 5.1 onwards, the location of TemporaryDirectory is a configuration setting of the 1E Client.

Function	**HttpGetFile**

The sub-directory and file is deleted when the instruction ends. If you want to keep the file, make the instruction copy it to a permanent location.

The following table shows the OS-specific defaults used if the setting is omitted, blank, or invalid. The default is recommended.

Platform	Environment Variable(s)	Default
Windows	TMP TEMP	*%windir%\Temp* which is the default *%TEMP%* location for SYSTEM.
Linux	TMPDIR	/tmp

Nomad integration

If 1E Nomad is available, it will optimize the download, storing a local copy of the file in the Nomad cache, with cache priority 1 and is subject to normal Nomad cache management.

Parameters

URL (string): Either (a) the relative path to be appended to the "*Content*" directory under the Background Channel URL, or (b) a full URL starting with "*http*" or "*https*".

It is a runtime error if the status code from a GET request on the full absolute URL is not 200.

Note

"*Content*" is created as a virtual folder in IIS during installation of the Tachyon platform.

Note

When an **absolute URL** is specified and the schema is '**https**' then the endpoint requesting the resource will attempt to verify the entire certificate chain of the resource as normal. This can fail if the endpoint does not recognize all the links in the chain of trust. The rules of checking are more rigorous than some browsers, which are often relaxed about the very root of the chain if it is signed using a **sha1** algorithm. This means whilst a browser may download the resource and display it, the

endpoint might actually complain about the validity of the root Certificate Authority certificate.

To resolve this, you need to get hold of the public key for the root certificate and:

- On Windows, it must be placed into the local machine trusted root certificate store
- On macOS, it needs to be in the 'System' folder of the Key Chain and explicitly trusted
- On other Unix/Linux-based endpoints that use a cacert.pem file, the root certificate should be exported as a base64-encoded CER format file and then the text appended to the cacert.pem file and the Agent restarted.

Hash (string): The SHA256 hash of the file. If the downloaded file's hash does not match this, a runtime error occurs.

Size (integer) : The size of the file in bytes. If the downloaded file's size does not match this, a runtime error occurs.

Suffix (string, optional): Adds the specified suffix to the temporary file name. Usually this is not necessary, but some applications require a filename to end with a certain suffix. If the specified suffix does not start with a '.' then one is added automatically. Cannot be used with the *PreserveName* parameter.

Function	**HttpGetFile**

Stagger (boolean, optional, default:true): Instructs the download operation to either adhere to the configured random execution stagger setting if true, or cause it to download immediately (so no staggering) if set to false. If the agent is running on Windows AND Nomad is installed and available AND it is decided by other means that Nomad will be used, then staggering will not come into effect, regardless of the value, because Nomad will take care of not overloading the network.

PreserveName (boolean, optional, default:false): If false the directory will be the system temporary setting and the filename will be random. If true then directory will be some unique space for the execution of the containing instruction and the stem and suffix in the *URL* parameter are used as the filename. Cannot be used with the *Suffix* parameter.

Return values

FilePath (string): The full path of the downloaded file.

Warning

This is a temporary file that persists only while the instruction is live. When the instruction ends the file will

Function	**HttpGetFile**

be deleted. If you want to keep the file, make the instruction copy it to a permanent location.

Example

A common use for this function is to download a script which can then be executed locally.

Example - download and execute a script

```
/* Download the script. Since we specify a partial UR
L (just the filename),
   the full URL is built by combining the specified U
RL with the root
   of the Tachyon Background channel. Note that becau
se HttpGetFile
   is a built-in function, there is no need to specif
y a Module name */
@resource1 = HttpGetFile(URL:"MyCustomScript.ps1",
                         Size:2137,
                         Hash:"e0b5bb08b68ed4cdd73d03
5a775458e34bce1d68ec93a11defa62ca25e3d7037");

/* @script now contains a single row with one column
named FilePath, so can
   be passed directly to the Scripting module */
@result = Scripting.Run(Language: "PowerShell", Scrip
t: @resource1, InterpretAsJson: true);
```

To download content from an external web-server, you can instead set the URL property to the full URL of the file to be downloaded.

Example - downloading content from an external URL

```
/* Download file from web server */
@resource1 = HttpGetFile(URL:"http://webserver.acme.c
om/MyFile.jpg",
                         Size:73355,
                         Hash:"c9411ef157a1b9e7d8418d
0d9297b8df3f633b6a1b2250d735c9b5cd32f1f795");

/* Build command line to copy temporary downloaded fi
le to final location */
@cmd = SELECT "cmd.exe /c copy " || FilePath || " C:\
\Pictures\\MyFile.jpg" AS CommandLine
       FROM @resource1;

/* Run the command built above */
NativeServices.RunCommand(CommandLine: @cmd.CommandLi
```

Function	**HttpGetFile**
	ne); /* Downloaded file will now be in C:\Pictures\MyFile.jpg */
Platforms	• All
Notes	

1.6.2.2 Raise

Warning

This function is for internal use only.

Function	**Raise**
Action	A built-in function to raise one or more events.
Parameters	*Topic* (string): The topic that will be applied to the supplied event(s), for example: "*Process.Crash*". **Note** The topic name is not case-sensitive. *Timestamp* (string, optional, default empty): A string representing the ISO-8601 timestamp that will be applied to the supplied event(s). **Note** If not given, the timestamp is the time of execution. *EventData* (table, optional, default empty): Data to be associated with the event(s) which will be raised. An event will be

Function	**Raise**
created for each row in the table. **Note** It is not mandatory for an event to have associated data, in which case just 1 event will be raised. *`Priority`* (string, optional, default 0): The priority of the event. *`0`* = standard, *`1`* = urgent. Standard events may be batched and not sent for a few seconds for efficiency reasons, whereas urgent events are sent immediately. New in v5.2.	
Return values	*`EventCount`* (integer): The number of events that have been successfully raised. **Note** This will be 1 if there was no *`EventData`* table or it was empty, else this will equal the number of rows provided in the *`EventData`* table for which an event was successfully raised.
Example	**Example - raise an event for each row in @stoppedServices** `Raise(Topic: "Service.Stopped", EventData: @stoppedServices, Timestamp: "2019-09-15T18:25:43.511Z");`
Platforms	• All
Notes	From v5.0.

1.6.2.3 IsUndefined

Function	**IsUndefined**

Action

Given a table as a parameter, this built-in function will output a table that indicates whether the input table is real and full of values or whether it is the result of an operation that produced nothing, which is different to producing an empty table.

Parameters

Table (table) The table to analyze.

Return values

A table with a single row and a single column called *Value* which will be set to one of the following:

Value	Meaning
1	The input table was undefined (no columns or rows)
0	The input table was defined (has columns at least)

Example

An instruction that generates and tests for an undefined table. The reason it is undefined is because the body of the foreach is never executed. The script output is column Value with a cell containing the number one.

```
@sequence = Utilities.GenerateSequence(Limit: 5);
@emptyTable = SELECT * FROM @sequence WHERE Sequence = 1942;
@undefinedTable = FOREACH @number IN @emptyTable DO
    SELECT * FROM @number WHERE Sequence > 3;
DONE;
@isUndefined = IsUndefined( Table : @undefinedTable );
SELECT * FROM @isUndefined;
```

An instruction that generates and tests for an defined table. The script output is column Value with a cell containing the number zero.

```
@sequence = Utilities.GenerateSequence(Limit: 5);
@definedTable = FOREACH @number IN @sequence DO
    SELECT * FROM @number WHERE Sequence > 3;
DONE;
@isUndefined = IsUndefined( Table : @definedTable );
SELECT * FROM @isUndefined;
```

Platforms

- All

Notes

Only works for @tables not $tables.

1.6.3 Tachyon Agent Built-in Keywords

Alphabetic listing of Agent built-in keywords.

The Agent language includes keywords to aid with instruction flow control. These are not **case-sensitive**. See Agent built-in keywords table on the Functions and Methods Reference page to know the earliest version of the Agent in which a built-in keyword was implemented.

Agent Built-in Keyword groupings

- ERROR— Immediately terminate an instruction with exit code "Error" with a custom error message.
- EVALUATE— Immediately terminate an instruction with exit code "Success, no content" if the preceding statement returned no results.
- FOREACH IN DO DONE— Allows you to repeat a block of one or more statements for each row in a @table, and optionally save in a @table the output of the last statement in the block.
- IF ELSE ENDIF— Allows conditional branching of instruction flow based on a condition.
- NOCONTENT— Immediately terminate an instruction early with an exit code “Success, no content”.
- NOTIMPLEMENTED— Immediately terminate an instruction with exit code "Not implemented" and a custom error message.
- RETURN— Immediately terminate an instruction with a given set of results.
- true false— Used as parameter values for certain Agent methods.

1.6.3.1 ERROR

Keywords	**ERROR**
Purpose	Immediately terminate an instruction with exit code "Error" with a custom error message.

Description	**Syntax**
	`//some statements`
	`ERROR "custom error message";`
	ERROR causes an instruction to terminate with exit code of 2 meaning “Error” and the specified custom error message. The error message must be supplied as a string literal, and cannot be a reference to a @table or any other kind of expression.

Keywords	**ERROR**

Examples	`@svcs = OperatingSystem.GetServiceInfo();`
	`@nomad = SELECT * FROM @svcs WHERE Name = "NomadBranch";`
	
	`IF NOT (@nomad)`
	`  ERROR "1E Nomad is not available";`
	`ENDIF;`
	
	`// Otherwise do some work with 1E Nomad...`

Notes	See SCALE User Guide: Exit Codes.

1.6.3.2 IF ELSE ENDIF

Keywords	**IF ELSE ENDIF**
Purpose	Allows conditional branching of instruction flow based on a condition.

Description	**Syntax**
	
	`IF (condition)`
	`  //IF block of statements: SELECT statements, Methods and/or Functions`
	`  //optional 'early exit' termination`
	`ELSE`
	`  //optional ELSE block of statements: SELECT statements, Methods and/or Functions`
	`  //optional 'early exit' termination`
	`ENDIF;`
	
	The condition cannot be a SELECT statement, but can be one of the following:
	
	• A variable, such as "*@myTable*"
	• A method invocation, such as "*Network.GetConnections()*"
	• A literal, such as "1" (although use of this is uncommon)
	
	In each of the above cases, the condition (which represents a table of data) will evaluate to "true" – and will therefore pass the IF check – if that table has one or more rows. A table with zero rows (including tables which have an undefined structure) will evaluate to "false".
	
	The IF condition is followed by a block of code to execute if the condition evaluates to true, and then a closing *ENDIF*. You may optionally include an *ELSE*

Keywords	**IF ELSE ENDIF**
	statement with a corresponding block of code to execute if the condition evaluates to false. Take care to ensure the output of the IF and ELSE blocks have the same shape, as described in SCALE User Guide: Inconsistent branching results.
Examples	See SCALE User Guide: Conditional branching using IF/ELSE/ENDIF.
Notes	

1.6.3.3 RETURN

Keywords	**RETURN**
Purpose	Immediately terminate an instruction with a given set of results.

Description	**Syntax**
	
	`@results = "some statement";`
	`RETURN @results;`
	
	Ordinarily, the results of an instruction are based on the last statement to execute; RETURN allows you to do this at any point in an instruction. In contrast to EVALUATE (which means “break out of this instruction if the preceding statement returned no results), *RETURN* breaks unconditionally. With *RETURN*, you supply the @table whose value should become the results of the instruction.
	
	The instruction exits successfully, with an exit code of 0 if the results have content, and 1 if they have no content.
	
	Take care to ensure the results returned have the same shape as the results returned by other parts of the instructions, as described in SCALE User Guide: Inconsistent branching results.

Examples	`@reg = NativeServices.RegistryGetValue(`
	`  Hive:"HKLM",`
	`  Subkey:"SOFTWARE\\MyVendor\\MyConfiguration",`
	`  Name:"SecuritySetting");`
	
	`// If the setting exists, return a message indicating we're secure`
	`IF (@reg)`
	`  @msg = SELECT "System already secure" AS Status;`
	`  RETURN @msg;`
	`ENDIF;`

Keywords	**RETURN**

```
// Otherwise, take some action...
// ... and report that back. No need to use RETURN, because
// this is the last statement in the instruction.
SELECT "System was insecure, but is now secure" AS Status;
```

Notes See SCALE User Guide: Exit Codes.

1.6.3.4 EVALUATE

Keywords	**EVALUATE**

Purpose Immediately terminate an instruction with exit code "Success, no content" if the preceding statement returned no results.

Description **Syntax**

```
//SQL statements
EVALUATE;
```

EVALUATEcauses an instruction to terminate with exit code of 1 meaning “Success, no content” if the preceding statement returned no results.

EVALUATE is the simplest form of flow control, and allows you to optimize your instruction and improve its readability, by avoiding unnecessary execution of subsequent statements.

Examples See SCALE User Guide: Using "EVALUATE" to break on no output.

Notes

1.6.3.5 FOREACH IN DO DONE

Keywords	**FOREACH IN DO DONE**

Purpose Allows you to repeat a block of one or more statements for each row in a @table, and optionally save in a @table the output of the last statement in the block.

Description **Syntax**

```
FOREACH @newTable IN @existingTable
DO
   //block of statements using @newTable (SELECT statements, Metho
ds and/or Functions)
```

Keywords	**FOREACH IN DO DONE**

```
   //optional 'early exit' termination
DONE;
```

FOREACH does not have any conditional flow control (eg. WHILE or BREAK) but this may change in a future version of Tachyon. However, you can cause the instruction to terminate early using EVALUATE, RETURN, NOCONTENT, ERROR, and NOTIMPLEMENTED..

Examples — See SCALE User Guide: Looping with FOREACH.

Notes — SeeSCALE User Guide: Inconsistent branching results.

1.6.3.6 NOCONTENT

Keywords	**NOCONTENT**

Purpose — Immediately terminate an instruction early with an exit code “Success, no content”.

Description — **Syntax**

```
//some statements
NOCONTENT;
```

NOCONTENT is an unconditional version of the *EVALUATE* keyword. Each causes an instruction to exit successfully, with an exit code of 1 meaning “Success, no content”, and do not return a message.

Examples

```
@users = Users.GetLoggedOnUsers();

// Return "success no content" if they are any logged in users
IF (@users)
    NOCONTENT;
ENDIF;

// ... otherwise do something which can only be done if no-one is
logged on
```

Notes — See SCALE User Guide: Exit Codes.

1.6.3.7 NOTIMPLEMENTED

Keywords	**NOTIMPLEMENTED**
Purpose	Immediately terminate an instruction with exit code "Not implemented" and a custom error message.

Description	**Syntax**
	`//some statements`
	`NOTIMPLEMENTED "custom error message";`
	NOTIMPLEMENTED causes an instruction to terminate with exit code of 3 meaning “Not implemented” and the specified custom error message. The error message must be supplied as a string literal, and cannot be a reference to a @table or any other kind of expression.

Examples	`@dev = Device.GetSummary();`
	`@win = SELECT * FROM @dev WHERE OperatingSystemType = "Windows";`
	
	`IF NOT (@win)`
	`  NOTIMPLEMENTED "This functionality is only available on Windows";`
	`ENDIF;`
	
	`// ... otherwise do something Windows-specific`

Notes	See SCALE User Guide: Exit Codes.

1.6.3.8 true false

Keywords	**true false**
Purpose	Used as parameter values for certain Agent methods.
Description	*true* and *false* are keywords used as parameter values for Agent methods that required data-type 'bool'. Bool type parameter values do not accept @variables. These keywords are not case-sensitive, but unlike other keywords, the convention is to use lowercase.
Examples	*FileContent.FindAndAppend(FilePath: "c:\\temp\\queen.txt", FindString: "i want to break free!", AppendString: "I want to break free!", Backup: true, CaseSensitive: false);* *Tagging.Get(Name: "WithScanner", Scopable: false);*

Keywords	**true false**
	NativeServices.RunCommand(CommandLine: "ping 127.0.0.1", SplitLines:true, IgnoreExitCode:true);
Notes	True and False cannot be used in SQLite SELECT statements, which must use 1 and 0 instead.

1.6.4 Tachyon Agent SQL Functions

Alphabetic listing of Agent SQL functions. Under this page are separate pages with full details for each function.

In addition to SQLite functions, the Agent language takes advantage of SQLite extensibility by providing "application defined functions" for use in SELECT statements. SQL keywords and function names are not case-sensitive, but convention is to write them in UPPERCASE.

See the Agent SQL functions versions table on the Functions and Methods Reference page to know the earliest version of the Agent in which a SQL function was implemented.

1.6.4.1 BASE64DECODE

Function	**BASE64DECODE**
Action	Produces a decoded Base64/Base64Url string.
Parameters	(string) Input Base64/Base64Url text to be decoded.
Return values	(string) Decoded Base64/Base64Url string.

Example	`// Returns "ABCabc123!@#=|>z<?"`
	`SELECT BASE64DECODE("QUJDYWJjMTIzIUAjPXw+ejw/") AS Base64Decoded;`
	`// Returns "ABCabc123!@#=|>z<?"`
	`SELECT BASE64DECODE("QUJDYWJjMTIzIUAjPXw-ejw_") AS Base64UrlDecoded;`

Notes	Introduced in v23.5 From v24.9 onwards, this function also decodes a Base64UrlEncoded string

1.6.4.2 BASE64ENCODE

Function	**BASE64ENCODE**
Action	Produces a Base 64 encoded string for the input text.

Function	**BASE64ENCODE**
Parameters	(string) Input text to be encoded.
Return values	(string) Base 64 encoded string.
Example	`// Returns "QUJDYWJjMTIzIUAjPXw+ejw/"` `SELECT BASE64ENCODE("ABCabc123!@#=|>z<?") AS Base64Encoded;`
Notes	Introduced in v23.5

1.6.4.3 BASE64URLENCODE

Function	**BASE64URLENCODE**
Action	Produces a Base 64 URL encoded string for the input text.
Parameters	(string) Input text to be encoded.
Return values	(string) Basename (filename with file extension if present).
Example	`// Returns "QUJDYWJjMTIzIUAjPXw-ejw_"` `SELECT BASE64URLENCODE("ABCabc123!@#=|>z<?") AS Base64UrlEncoded;`
Notes	Introduced in v24.9

1.6.4.4 BASENAMEFROMPATH

Function	**BASENAMEFROMPATH**
Action	Takes a path as a string and returns the basename.
Parameters	(string) Full path and filename.
Return values	(string) Basename (filename with file extension if present).
Example	`// Returns "file.txt"` `SELECT BASENAMEFROMPATH("c:\\tmp\\things\\file.txt") AS Basename;`
Notes	The function is not file system aware; it does not check to see if the path of file exists.

Function	**BASENAMEFROMPATH**
	The function is OS aware; it assumes the directory separator is \ on Windows and / on other OS. Like the UNIX basename command, it returns everything after the last path separator character in the string supplied, however this version does not support removing the suffix (extension) from the basename.

1.6.4.5 COMBINEPATHS

Function	**COMBINEPATHS**
Action	Takes two paths as strings, or a path and a basename as strings. It combines them and returns the result.
Parameters	(string) Path. (string) Path, filename, or path and filename.
Return values	(string) Concatenation of input parameters with a directory separator between them.

Examples	`// Returns "c:\tmp\things\file.txt" on Windows OS`
	`SELECT COMBINEPATHS("C:\\tmp","things\\file.txt") AS Path;`
	
	`// Returns "/var/log\things/file.txt" on Windows OS`
	`// Returns "/var/log/things/file.txt" on non-Windows OS`
	`SELECT COMBINEPATHS("/var/log","things/file.txt") AS Path;`

Notes	The function is not file system aware; it does not check to see if the path of file exists, and does not change the input strings. The function is OS aware; it assumes the directory separator is \ on Windows and / on other OS.

1.6.4.6 COMPAREVERSIONS

Function	**COMPAREVERSIONS**
Action	Compare two version strings.
Parameters	(string) Version string 1 of the form major[.minor[.release[.build]]] where "7.3.2.8", "5.7.1", "2.29" and "14" are all valid version strings.

Function	**COMPAREVERSIONS**
	(string) Version string 2 of the formmajor[.minor[.release[.build]]]
Return values	(integer) An integer value representing the result of the first version string relative to the second: -4 If the first version string's major is lower than the second version string's -3 if the first is lower but major is the same -2 if the first is lower but major.minor is the same -1 if the first is lower but major.minor.release is the same 0 if the same 1 if the first is higher but major.minor.release is the same 2 if the first is higher but major.minor is the same 3 if the first is higher but major is the same 4 if the first's major is higher **Note** If an invalid version string is supplied then a null value is returned. "Missing" fields are treated as numerically 0.

Examples	`@a = Software.GetInstallations();`
	`@b = select * from @a where Product like "Oracle VM VirtualBox";`
	`select * from @b WHERE compareversions(@b.version, "5.1.2.14" ) < 0;`
	Finds VirtualBox packages that are older than 5.1.2.14.

1.6.4.7 DATETRUNC

Function	**DATETRUNC**
Action	Truncates a date-time value, expressed as a Unix epoch time integer, to a given resolution and returns a corresponding Unix epoch time integer.
Parameters	(integer) Unix epoch time integer, representing a datetime value (string) Resolution - Extent to which the date time should be truncated, which can be one of:

Function	**DATETRUNC**

-
  - "year" or "y"
  - "month" or "m"
  - "day" or "d"
  - "hour" or "H"
  - "minute" or "M"

Return values

(integer) Truncated date time value expressed as a Unix epoch time.

Example

```
// Input is 1492761464 == 21 April 2017 07:57:44 UTC
@x = SELECT 1492761464 AS Input;

// Returns 1483228800 == 01 January 2017 00:00:00 UTC
SELECT DATETRUNC(@x.Input, "year") AS Truncated FROM @x;

// Returns 1492758000 == 21 April 2017 07:00:00 UTC
SELECT DATETRUNC(@x.Input, "hour") AS Truncated FROM @x;
```

The following is an example of truncating 18-August-2017 19:18:03:

```
@datetime = SELECT "2017-08-18T19:18:03Z" AS string;
// convert date string to unixepoch integer, truncate, then conve
rt result back to ISO-8601 formatted string
SELECT EPOCHTOJSON( DATETRUNC( STRFTIME("%s",string), "H") ) AS T
S FROM @datetime;
```

truncation	ISO-8601 result
"M" minute	2017-08-18T19:18:00Z
"H" hour	2017-08-18T19:00:00Z
"d" day	2017-08-18T00:00:00Z
"m" month	2017-08-01T00:00:00Z
"y" year	2017-01-01T00:00:00Z

For further examples, see datetime handling.

Function	**DATETRUNC**
Notes	Note that this function performs truncation, as opposed to rounding.

1.6.4.8 DIRECTORYFROMPATH

Function	DIRECTORYFROMPATH
Parameters	(string) Path of file or directory.
Return values	(string) Path.

Example	`@path = "c:\\tmp\\things\\files\\file.txt";`
	`// returns c:\tmp\things\files`
	`@path = SELECT DIRECTORYFROMPATH(Value) AS Value FROM @path;`
	`// returns c:\tmp\things`
	`@path = SELECT DIRECTORYFROMPATH(Value) AS Value FROM @path;`
	`// returns c:\tmp`
	`@path = SELECT DIRECTORYFROMPATH(Value) AS Value FROM @path;`
	
	`// This example returns "/var/log/product2"`
	`SELECT DIRECTORYFROMPATH("/var/log/product2/info.log");`

Notes	The function is not file system aware; it does not check to see if the path or file exists. The function is OS aware; it assumes the directory separator is \ on Windows and / on other OS. Like the UNIX dirname command, it returns everything before the last path separator character in the string supplied.

1.6.4.9 EPOCHTOJSON

Function	EPOCHTOJSON
Parameters	(integer) Unix epoch time in numeric form.
Return values	(string) The time as an ISO-8601 string.

Examples	`// Input is 1492761464 == 21 April 2017 07:57:44 UTC`
	`// Returns "2017-04-21T07:57:44.000Z"`
	`SELECT EPOCHTOJSON(1492761464) AS MyDateTime;`

Function	EPOCHTOJSON
Notes	This function is useful as all date-times returned by the Tachyon Agent must be in ISO-8601 format to be parsed correctly by the Tachyon Server. Seehttps://www.epochconverter.com/ For more examples, see Datetime handling.

1.6.4.10 EXPAND

Function	**EXPAND**
Parameters	(string) A string containing environment variables in the form *%variable%* for Windows, and *$variable* for non-Windows operating systems.
Return values	(string) The string with the environment variables replaced, where possible.

Example	Returns "*xcopy C:\Windows\MEMORY.DMP \\share\crashdumps*":
	
	`SELECT "xcopy " || EXPAND("%windir%\\MEMORY.DMP") || " \\\\share\\crashdumps" AS CommandLine;`
	
	This Linux example shows how a variable that does not exist is returned unchanged, returning the "*$IDONOTEXIST does not exist*" as the value of *ExpandedText*:
	
	`SELECT EXPAND("$IDONOTEXIST does not exist") AS ExpandedText;`

Notes	Replacements will only occur for environment variables. If an environment variable does not exist, *Expand* will not modify that portion of the string. (This is different from certain shells such as Bash which usually expand an undefined variable to an empty string.) If an empty string is supplied as the parameter, an empty string will be returned. If a NULL is supplied as the parameter, a NULL will be returned. On non-Windows, daemons (including the Tachyon Agent) have a limited set of environment variables compared with user accounts.

1.6.4.11 HASHSTRING

Function	**HASHSTRING**
Action	Produce a hash of a string value based on a named hashing algorithm
Parameters	(string) Hash algorithm; one of "MD5", "SHA1" or "SHA256" (string) Input data
Return values	(string) Hash of input data using the named algorithm, expressed as a (lowercase) hexadecimal string

Example	`// Show hash values using all three supported algorithms for two different sample data inputs, an empty string, and null`
	`@values = SELECT "My sample data" AS Val UNION ALL SELECT "My other sample data" UNION ALL SELECT "" UNION ALL SELECT NULL;`
	`@algos  = SELECT "MD5" AS Algo UNION ALL SELECT "SHA1" UNION ALL SELECT "SHA256";`
	
	`SELECT  IFNULL(V.Val, "(null)") AS Input_Value`
	`,       A.Algo AS Input_Algo`
	`,       IFNULL(HASHSTRING(A.Algo, V.Val), "(null)") AS Output_Hash`
	`FROM    @values as V`
	`,       @algos AS A;`

Notes	Note that *`HASHSTRING(algorithm, NULL)`* will always yield NULL, whereas *`HASHSTRING(algorithm, "")`* (i.e. hash of empty string) will produce a corresponding hash value. Passing a non-string value, such as a number, as the input data will cause the input string to be converted to its default textual representation before being hashed, so *`HASHSTRING(algorithm, 123)`* will yield the same value as *`HASHSTRING(algorithm, "123")`*. You should, however, avoid this behaviour where possible and prefer to pass string data. This is because the implicit string conversion may yield an unexpected result, such as hashing constants e.g. true or false (which are represented internally as 0 and 1). **New in 8.1**

1.6.4.12 IPINRANGE

Function	**IPINRANGE**
Action	Determines if an IP Address is within a specified range or ranges.

Function	**IPINRANGE**
Parameters	*(String, mandatory):* **IpAddress**. The IPv4 address in print notation, for example, "192.168.0.1". *(String, mandatory):* **IpRange**. This is the IPv4 range that provides the boundaries for the range checking. Multiple ranges can be supplied, with a comma as a separator. An IpRange is two IP Addresses separated by a hyphen. For example: "192.168.0.0-192.168.255.255,1.2.3.4-2.3.4.5" *(Integer, optional):* **AllowNullAddresses**. This can be 0 or 1. Set to 1 if a NULL is supplied as parameter 1, then it will cause the function to return 1 (true). This acts as a passthrough for sets of data where only some rows will have an IpAddress, but the other data is still valuable. In effect, only IPAddresses which exist will be range checked, the rest will be allowed through. Possible values are 1, (allow null IPs) or 0 (do not allow null IPs). The default is 0. *(Integer, optional)*: **AllowIpv6Addresses**. This can be 0 or 1. Set to 1 if an IPv6 address is supplied as IpAddress and/or IpRange, then it will cause the function to return 1 (true). Similar to parameter 3, this will allow through IPv6 Addresses, but will continue filtering IPv4 addresses. Possible values are 1, (allow IPv6 addresss) or 0 (do not allow IPv6 addresses). The default is 0. **Note** If any optional parameters need to be specified, then all optional parameters must be specified. As a result the number of arguments to this function is either 2 or 4.
Return values	(Integer) True (1) or false (0).

Examples	`// Input is "192.168.1.1", which is checked against the ranges "1.2.3.4-2.3.4.5" and "192.168.0.0-192.168.2.2"`
	`// Returns 1 (true)`
	`SELECT IPINRANGE("192.168.1.1","1.2.3.4-2.3.4.5,192.168.0.0-192.168.2.2");`
	
	`// Input is "192.168.1.1", which is checked against the ranges "1.2.3.4-2.3.4.5" and "192.168.0.0-192.168.2.2".`
	`// The optional parameters are set to 0, their defaults. This is functionally the same as the above example.`
	`// Returns 1 (true)`
	`SELECT IPINRANGE("192.168.1.1","1.2.3.4-2.3.4.5,192.168.0.0-192.168.2.2",0,0);`
	
	`// This will return 0 (false) - the IP is not in the range`
	`// Returns 0 (false)`

Function	**IPINRANGE**

```
SELECT IPINRANGE("192.168.66.77,"192.168.0.0-192.168.66.76");

// This allows the following IP through even though it is an IPv6
address and thus not range checked
// Returns 1 (true)
SELECT IPINRANGE("::1","1.2.3.4-2.3.4.5",0,1);

// This allows the following IP though even though it is NULL and
thus cannot be range checked
// Returns 1 (true)
SELECT IPINRANGE(NULL,"1.2.3.4-2.3.4.5",1,0);

// This will still return 1 (true) - IPv6 ranges are discarded (th
ough the agent will log a warning)
// Returns 1 (true)
SELECT IPINRANGE("192.168.1.1","::1-::55",192.168.0.0-192.168.2.2
");

// This will return 0 (false) - the ranges cannot be correctly par
sed. Though ::1-::55 would be discarded and a
// warning will be logged, jaffa-cake will cause a delicious erro
r. This won't stop the operation, it'll just
// return 0 instead.
// Returns 0 (false)
SELECT IPINRANGE("192.168.1.1","::1-::55",jaffa-cake");

// More meaningful usage: select everything from the $TCP_Live his
toric table with an IPv4 address within a range
@raw = SELECT * from $TCP_Live;
SELECT * from @raw where IPINRANGE(@raw.IpAddress,"192.168.0.0-19
2.168.255.255");
```

Notes

1.6.4.13 IPV4STRTOINT

Function	**IPV4STRTOINT**
Action	Converts an IPv4 address into an integer.
Parameters	(string) IPv4 Address in print notation, for example "10.20.30.45"
Return values	(integer) The IPv4 Address represented as an integer, for example 169090605 = (256^3 * 10) + (256^2 * 20)+ (256 * 30) +(45)

Function	**IPV4STRTOINT**

Examples	`// Input is "10.20.30.45"`
	`// Returns 169090605`
	`SELECT IPV4STRTOINT("10.20.30.45") AS Integer;`

Notes	This function is useful when comparing IP Addresses. Does not support IPv6.

1.6.4.14 NEWID

Function	**NEWID**
Action	Generates a new universally unique identifier (also known as a GUID)
Parameters	None
Return values	(string) New identifier of the form "*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx*".

Example	`// Generate a list of 10 random GUIDs`
	`@seq = Utilities.GenerateSequence(Limit: 10);`
	`SELECT NewId() AS MyNewGuid FROM @seq;`

Notes	Note that the return value is a string, and is returned in lowercase. **New in v8.1**

1.6.4.15 REGEXP

Function	**REGEXP**
Action	Determines if a string matches a regular expression.
Parameters	(string) Regular expression in ECMA262 format. (string) Input string.
Return values	(integer) Match is true (1) or false (0).

Examples	`// Example data`
	`@testTable = SELECT "This will pass the test" AS Data`
	`UNION SELECT "This will not pass the test."`
	`UNION SELECT "this won't pass the test";`
	
	`// This example uses the REGEXP function to identify which strings`

Function	**REGEXP**

```
in the test table match and do not match
SELECT Data, REGEXP("This .* test", Data) AS [Match] FROM @testTab
le;

// This example uses the REGEXP operator to return only matching s
trings
SELECT Data FROM @testTable WHERE Data REGEXP "This .* test";
```

Notes

Uses ECMA262 format for regular expressions:
http://www.cplusplus.com/reference/regex/ECMAScript/
http://en.cppreference.com/w/cpp/regex/ecmascript

REGEXP can be used as a both a function, and also as an operator, as explained in www.sqlite.org/lang_expr.html#regexp.

```
// Function
REGEXP("<regular expression>", "<input string>")

// Infix operator
"<input string>" REGEXP "<regular expression>"
```

Note

Backslashes inside the REGEXP expression will need additional escaping:

```
@test = select "c:\\users\\potato\\documents\\business" as Path;
// Note the additional escaping below inside the REGEXP(expr,dat
a):
select * from @test where REGEXP("c:\\\\users\\\\.*\\\\documents\\
\\business", Path);
```

1.6.4.16 REGEXPCAPTURE

Function	REGEXPCAPTURE

Parameters

(string) Regular expression.

(string) Input string.

Return values

(string) Captured string.

Examples

```
// This example returns "something is better than nothing"
SELECT REGEXPCAPTURE("something (.*) nothing", "People say that so
mething is better than nothing, but they're wrong!") AS Capture;

// This example returns rows from @commandLines, and pulls out the
"parentProcessId" command-line argument
```

Function	REGEXPCAPTURE
	`SELECT REGEXPCAPTURE("/parentProcessId .* ", CommandLine) AS ParentProcessIdArgument FROM @commandLines;`
Notes	Uses ECMA262 format for regular expressions: http://www.cplusplus.com/reference/regex/ECMAScript/ http://en.cppreference.com/w/cpp/regex/ecmascript

1.6.4.17 REGEXPREPLACE

Function	**REGEXPREPLACE**
Action	Uses a regular expression to replace a string.
Parameters	(string)Regular expression in ECMA262 format to be used for matching; enclosing an expression in parentheses will cause the value for that expression to be "captured" and used in the replacement string. (string) Input string to which the Regular expression will be applied. (string) Replacement string, using $1, $2, etc. as placeholders for sub-matches. See for example https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace#Specifying_a_string_as_a_parameter.
Return values	(string) The original string with all the matching text replaced with the replacement text.

Examples	`// This example returns "blue and red"`
	`SELECT REGEXPREPLACE("(.*) and (.*)", "red and blue", "$2 and $1");`
	
	`// This example takes rows from @iniFileLines, and splits the INI-file style lines`
	`// (e.g. MyKey=MyValue) into Key and Value columns`
	`// Note the "?" in the regular expression to indicate a non-greedy match`
	`SELECT REGEXPREPLACE("(.*?)=(.*)", Line, "$1") AS Key`
	`,      REGEXPREPLACE("(.*?)=(.*)", Line, "$2") AS Value`
	`FROM   @iniFileLines;`
	
	`// This example returns "747", i.e. the second (out of three) captures`
	`SELECT REGEXPREPLACE("(.*?)([0-9]+)(.*)", "The 747 is my favourite plane", "$2") AS ModelNumber;`

Notes	Uses ECMA262 format for regular expressions: http://www.cplusplus.com/reference/regex/ECMAScript/ http://en.cppreference.com/w/cpp/regex/ecmascript

1.7 Guaranteed State Triggers

1.7.1 Triggers

As well as the SCALE - which determines *what* is executed when the rule is processed - a rule has one or more associated triggers which determines *when* the rule is processed.

The Tachyon Agent supports the following triggers:

- AgentStartup — Triggers when Tachyon Agent starts.
- DirectoryChange — Triggers when the specified directory is modified (including created or deleted). If the corresponding directory does not exist at the point that the trigger is initialized, the trigger will poll periodically (approx. every minute) for this directory to be created.
- FileChange — Triggers when a nominated file is modified (including created or deleted). If the corresponding file does not exist at the point that the trigger is initialized, the trigger will poll periodically (approx. every minute) for this file to be created.
- Interval — Triggers periodically at the specified frequency (IntervalSeconds, IntervalMinutes or IntervalHours), allowing a rule to be evaluated at the given interval.
- ServiceStatusChange — Triggers when the state of e specified Windows service changes - when it is stopped, started, paused, resumed, installed or uninstalled.
- WindowsEventLog — Triggers when a log entry is created in a specific Event Log, optionally matching an XPath filtering expression. This can include the "Classic" Windows Event Logs (Application, Security, Setup and System) and also the "Applications and Services Logs" (although does not include the "Analytic and Debug" logs).
- WindowsRegistryChange — Triggers when a log entry is created in a specific Event Log, optionally matching an XPath filtering expression. This can include the "Classic" Windows Event Logs (Application, Security, Setup and System) and also the "Applications and Services Logs" (although does not include the "Analytic and Debug" logs).

Triggers are parameterized - for example, the "FileChange" trigger has a parameter named "FileName" which determines which file is to be monitored.

If a rule is associated with multiple triggers, that rule will be evaluated whenever ANY of the corresponding triggers fire. For example, a rule might have both an Interval trigger and a FileChange trigger, which might test the contents of a file when it is changed, and also periodically.

Additionally, a rule is evaluated automatically - irrespective of the associated triggers - when it is first received by the Tachyon Agent, and also when the Tachyon Agent starts up. This is to account for any events which might have occurred while the Tachyon Agent was offline that may have been missed.

AgentStartup trigger

Given the statement above, this raises the question of why there is an "AgentStartup" trigger at all.

Simply, this trigger is a "fake" trigger which allows creation of a rule which is only ever checked on Agent start-up, and is required because a rule must always have *at least one* associated trigger.

Each of the supported triggers is described in detail below. Before this, it is important to understand the concept of "de-bouncing".

1.7.2 Triggers vs Trigger Templates

When the Agent receives its policy, it is provided the trigger(s) for each corresponding rule, and the parameter values for each of those triggers.

However, on the Tachyon Server, there is the additional concept of Trigger Templates. A Trigger Template is a pre-configured variant of a specific Agent-based trigger, which itself may have parameters. Trigger Templates are presented to the user when creating a rule in the Guaranteed State user interface (and similarly must be specified when calling the Consumer API directly to create a rule).

In summary, a Trigger Template provides a way to simplify user selection of an event, without necessarily having to provide all the parameters for the Agent's trigger themselves.

When creating a rule, a user selects a Trigger Template - not a trigger itself.

Consider the following example:

- We may wish to have several rule evaluate whenever any of the settings for an item of software are changed in the registry, which can be achieved using the WindowsRegistryChange trigger
- This trigger, as per the earlier section, takes three parameters (Hive, Subkey, IncludeSubkeys)
- Rather than having the user type in "*HKLM*", "*SOFTWARE\Acme\SpreadsheetApplication*" and "*true*" for the corresponding parameter values for this trigger, it is preferable to have a "specialized" Trigger Template which contains these parameters pre-configured
- In this way, a user creating a rule could simply select a Trigger Template called "Whenever the settings for Spreadsheet Application change" and not have to provide any parameter values

To do this, a Trigger Template could be created which specializes the WindowsRegistryChange trigger.

A trigger template has the following properties:

Property	Description	Notes
Description	Textual description of the trigger	
Type	The corresponding trigger type for the Tachyon Agent to use (e.g. ServiceStatusChange)	
ReadablePayload	Textual description with placeholders for user-supplied parameters, which gets interpolated to form the description of the trigger	Similar to ReadablePayload for Fragments and Instruction Definitions

Property	Description	Notes
		Each user parameter's placeholder should appear at least once in the ReadablePayload
AgentParameters	The set of parameter name-and-value pairs to be sent to the Tachyon Agent, as per the "Parameters" section for each of the trigger types defined in the previous section	The "value" for each parameter can contain placeholders for user-supplied parameters, which will be interpolated before being sent to the Tachyon Agent
UserParameters	The set of parameters - if any - to be presented to the user when this trigger is selected	The values supplied by the user for each parameter can be injected into the parameters passed to the Tachyon Agent

To better understand the concept of Agent Parameters and User Parameters, we'll consider a second Trigger Template example, and then look at how the Trigger Templates for both examples might be defined.

For our second example:

- We may wish to have a rule evaluate whenever the configuration of a specific Windows Service is modified
- Again, we can use the WindowsRegistryChange trigger to achieve this by asking it to monitor changes under*HKLM\SYSTEM\CurrentControlSet\Services\<SomeService>*
- To avoid the user having to provide "*HKLM*" and "*SYSTEM\CurrentControlSet\Services*" each time, a Trigger Template could be created which simply prompts the user for the name of the service whose settings should be monitored
- This service name could then be "injected" into the "Subkey" parameter for the Agent's trigger

Here's the definition of the Trigger Template for our first example (monitoring changes to the Spreadsheet Application's registry):

Property	Value	Explanation
Description	When the settings for My Spreadsheet Application change	
Type	WindowsRegistryChange	This is the type of trigger that the Agent should use (i.e. check for registry changes)

Property	Value	Explanation
ReadablePayload	On change of My Spreadsheet Application settings	
AgentParameters	Hive=HKLM Subkey=SOFTWARE\Acme\SpreadsheetApplication IncludeSubkeys=true	These are the parameters for the trigger that the Agent will receive
UserParameters	(none)	The user is not required to enter any parameters to use this trigger template

Again - note that in that example, the user is not required to provide the values for any parameters in order to use that trigger template.

Here's the definition of the Trigger Template for the second example, where the user needs to enter the name of the service whose settings should be monitored:

Property	Value	Explanation
Description	When the settings for a Windows service change	
Type	WindowsRegistryChange	As per previous example
ReadablePayload	On change of settings for the ***%ServiceName%*** service	Note that in this case, the readable payload contains the substitution value for our "service name" parameter. When this Trigger Template is used and a value is supplied, e.g. "WinMgmt" (to monitor for changes in WMI service configuration), the user would see: *"On changes of settings for the WinMgmt service"*
AgentParameters	Hive=HKLM	These are the parameters for the trigger that the Agent will receive.

Property	Value	Explanation
	Subkey=SYSTEM\CurrentControlSet\Services\ ***%ServiceName%*** IncludeSubkeys=true	Note that the substitute value for the ServiceName user parameter occurs in the "Subkey" parameter; this will be interpolated at the point that it is sent to the Agents
UserParameters	ServiceName	We define a single parameter - the "ServiceName" which the user will be prompted to enter. The value supplied by the user will be injected into the resultant ReadablePayload and AgentParameters when it is associated with a policy rule. Note that the actual definition of a user parameter is more complex than just providing a name; user parameters follow the same convention as parameters for Instruction Definitions, and have additional attributes such as the data type, maximum length, "hint" text, etc. These additional attributes are not shown here for brevity.

1.7.2.1 Trigger de-bouncing

Some triggers - either implicitly (FileChange trigger) or explicitly (WindowsEventLog trigger) - employ the concept of "de-bouncing". This is a technique to prevent a trigger firing multiple times in rapid succession (and therefore causing a rule to be evaluated repeatedly) when, in practice, only a single "logical" event has happened.

Consider the case of the FileChange trigger: behind the scenes, this hooks into the Windows file monitoring API to determine when a file has been modified. However, when an application modifies a file, it might make a number of successive modifications in a very short space of time - for example, it might write a block of data, close the file, re-open the file, write another block of data, and then close the file. This manifests itself as a burst of events when really it should be seen as a single change.

To cater for this, the FileChange trigger automatically de-bounces. When a notification is received from the operating system to say that the file has changed, the Agent waits a small amount of time (in the case of the FileChange trigger, this is two seconds). If another notification is received within those two seconds, this wait is cancelled and a new two second wait is immediately started. When the two second period has elapsed and no

further notifications have been received from the operating system, the trigger is fired. In this way, the trigger waits for the notifications to "calm down" before it causes the associated rule to be evaluated.

Another example would be using the EventLog trigger to detect a change of network. If a laptop is unplugged from a wired Ethernet connection and switches to WiFi, there may be two successive "network change" events generated: "Wired network disconnected" followed shortly afterwards by "WiFi network connected". By employing de-bouncing (e.g. waiting for 10 seconds for network connectivity to stabilize), it is possible to view this as a single "Network connectivity has changed" event, rather than two independent events that occur one after the other (which would otherwise have caused the associated rule to be evaluated twice unnecessarily).

1.7.3 AgentStartup

Trigger	**AgentStartup**
Action	Triggers when Tachyon Agent starts.
Parameters	None.
Example	`<TriggerTemplate Name="AgentStartup" Category="Basic" Description="When the Tachyon Agent starts" Type="AgentStartup" ReadablePayload="On Tachyon Agent start-up" />`
Platforms	• All platforms
Notes	This is a pseudo-trigger; since all rules are evaluated on Agent start-up, this trigger provides no additional functionality. It can be used when no specific trigger is required, because a rule must always have at least one trigger.

1.7.4 DirectoryChange

Trigger	**DirectoryChange**
Action	Triggers when the specified directory is modified (including created or deleted). If the corresponding directory does not exist at the point that the trigger is initialized, the trigger will poll periodically (approx. every minute) for this directory to be created.
Parameters	*`DirectoryPath`* (string): The path of the directory to monitor for changes. Must be an absolute (non-network) path - e.g. "C:\Some Folder\My folder". Wildcards are not supported; the trigger supports monitoring a single directory only. However, the subdirectories will also be monitored.

Example	`<TriggerTemplate Name="DirectoryChange" Category="Event" Description="When a directory changes (Windows only)" Type="DirectoryChange" ReadablePayload="On change of directory "%directoryPath%"">`
	`  <AgentParameters>`
	`  <Parameter Name="DirectoryPath" Value="%directoryPath%" />`
	`  </AgentParameters>`

Trigger	**DirectoryChange**

```
  <UserParameterJson><![CDATA[[{"Name":"DirectoryPath","Pattern":
"%directoryPath%","DataType":"string","ControlType":"freeText", "C
ontrolMetadata":null,"Placeholder":"filename","DefaultValue":null,
   "Validation":{"Regex":null,"MaxLength":"1024","AllowedValues":n
ull,"NumValueRestrictions":null},"Value":null,"HintText":null,"Sou
rce":null}]]]>
  </UserParameterJson>
</TriggerTemplate>
```

Platforms

- Windows

Notes

Warning

Avoid monitoring one level down (eg c:\temp) as the trigger will also set up a monitor on the parent directory (in this case, c:\), as well, so that changes to the specified directory can be spotted, like it being created. Monitoring c:\ will be very intensive on the machine as every file operation the operating system does will trigger it.

1.7.5 FileChange

Trigger	**FileChange**

Action

Triggers when a nominated file is modified (including created or deleted). If the corresponding file does not exist at the point that the trigger is initialized, the trigger will poll periodically (approx. every minute) for this file to be created.

Parameters

FilePath (string): The path and name of the file to monitor for changes.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="FileChange" Category="Event" Description="W
hen a file changes (Windows only)" Type="FileChange" ReadablePaylo
ad="On change of file "%fileName%"">
  <AgentParameters>
    <Parameter Name="FilePath" Value="%fileName%" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"Name":"FileName","Pattern":"%file
Name%","DataType":"string","ControlType":"freeText","ControlMetada
ta":null,"Placeholder":"filename","DefaultValue":null,
  "Validation":{"Regex":null,"MaxLength":"1024","AllowedValues":nu
ll,"NumValueRestrictions":null},"Value":null,"HintText":null,"Sour
ce":null}]]]>
  </UserParameterJson>
</TriggerTemplate>
```

Trigger	**FileChange**
Platforms	• Windows
Notes	The filepath must an absolute (non-network) path - e.g. "C:\My Folder\MyFile.txt", not "MyFile.txt" or "\My Folder\MyFile.txt". Wildcards are not supported; the trigger supports monitoring a single file only. To monitor multiple files for a rule, attach multiple FileChange triggers.

1.7.6 Interval

Trigger	**WindowsEventLog**
Action	Triggers periodically at the specified frequency (IntervalSeconds, IntervalMinutes or IntervalHours), allowing a rule to be evaluated at the given interval.
Parameters IntervalSeconds (integer): The frequency of the trigger in seconds. Minimum value is 30 seconds; Maximum value is 604,800 seconds (i.e. one week) *IntervalMinutes* (integer): The frequency of the trigger in minutes. Minimum value is 1 minute; Maximum value is 10,080 minutes (one week). *IntervalHours*(integer): The frequency of the trigger in hours. Minimum value is 1 hour; Maximum value is 168 (one week).	**Note** Only one parameter is allowed. You must provide exactly one of IntervalSeconds, IntervalMinutes or IntervalHours.

Example	**IntervalHours**
	
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<TriggerTemplate Name="IntervalHours" Categ`
	`ory="Basic" Description="Periodic (hours)"`

Trigger	WindowsEventLog

```
Type="Interval" ReadablePayload="Every %int
ervalHours% hours">
  <AgentParameters>
    <Parameter Name="IntervalHours" Value=
"%intervalHours%" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"Name":"Int
ervalHours","Pattern":"%intervalHours%","Da
taType":"int","ControlType":"freeText","Con
trolMetadata":null,"Placeholder":"number of
hours","DefaultValue":"12",
"Validation":{"Regex":"[0-9]{1,3}","MaxLeng
th":"3","AllowedValues":null,"NumValueRestr
ictions":null},"Value":null,"HintText":nul
l,"Source":null}]]]></UserParameterJson>
</TriggerTemplate>
```

IntervalMinutes

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="IntervalMinutes" Cat
egory="Basic" Description="Periodic (minute
s)" Type="Interval" ReadablePayload="Ever
y %intervalMinutes% minutes">
  <AgentParameters>
    <Parameter Name="IntervalMinutes" Value
="%intervalMinutes%" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"Name":"Int
ervalMinutes","Pattern":"%intervalMinutes%
","DataType":"int","ControlType":"freeText
","ControlMetadata":null,"Placeholder":"num
ber of minutes","DefaultValue":"30",
"Validation":{"Regex":"[0-9]{1,5}","MaxLeng
th":"5","AllowedValues":null,"NumValueRestr
ictions":null},"Value":null,"HintText":nul
l,"Source":null}]]]></UserParameterJson>
</TriggerTemplate>
```

IntervalSeconds

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="IntervalSeconds" Cat
egory="Basic" Description="Periodic (second
s)" Type="Interval" ReadablePayload="Ever
y %intervalSeconds% seconds">
  <AgentParameters>
    <Parameter Name="IntervalSeconds" Value
="%intervalSeconds%" />
```

Trigger	**WindowsEventLog**
	</AgentParameters> <UserParameterJson><![CDATA[[{"Name":"IntervalSeconds","Pattern":"%intervalSeconds%","DataType":"int","ControlType":"freeText","ControlMetadata":null,"Placeholder":"number of seconds","DefaultValue":"3600","Validation":{"Regex":"[0-9]{1,6}","MaxLength":"6","AllowedValues":null,"NumValueRestrictions":null},"Value":null,"HintText":null,"Source":null}]]]></UserParameterJson> </TriggerTemplate>
Platforms	• All platforms
Notes	The Tachyon Agent does not persist the last time the Interval trigger was fired across a restart; if the Agent is restarted, the timer associated with the interval trigger is reset back to its initial time. **Warning** Take care when scheduling rules to be evaluated - scheduling a rule to evaluate too frequently can adversely affect endpoint performance. In general, scheduling a rule to be evaluated more frequently than a few minutes should be avoided. Where possible, use an event-based trigger (such as Event Log or Registry change) in preference to an interval-based one.

1.7.7 ServiceStatusChange

Trigger	**ServiceStatusChange**
Action	Triggers when the state of e specified Windows service changes - when it is stopped, started, paused, resumed, installed or uninstalled.
Parameters	ServiceName (string): The (short) name of the Windows Service whose status should be monitored. Example ServiceName "winmgmt" for "Windows Management Instrumentation" service (visible in the Windows Services control panel applet).

Trigger	**ServiceStatusChange**
Example	<?xml version="1.0" encoding="UTF-8"?> <TriggerTemplate Name="ServiceStatusChange" Category="Event" Description="When the state of the named Windows service changes" Type="ServiceStatusChange" ReadablePayload="On change of running state of the "%serviceName%" service"> <AgentParameters> <Parameter Name="ServiceName" Value="%serviceName%" /> </AgentParameters> <UserParameterJson><![CDATA[[{"Name":"ServiceName","Pattern":"%serviceName%","DataType":"string","ControlType":"freeText","ControlMetadata":null, "Placeholder":"short name of service (e.g. NomadBranch)","DefaultValue":null, "Validation":{"Regex":null,"MaxLength":"1024","AllowedValues":null,"NumValueRestrictions":null},"Value":null,"HintText":null,"Source":null}]]]> </UserParameterJson> </TriggerTemplate>
Platforms	• Windows Vista and later.
Notes	

1.7.8 WindowsEventLog

Trigger	**WindowsEventLog**
Action	Triggers when a log entry is created in a specific Event Log, optionally matching an XPath filtering expression. This can include the "Classic" Windows Event Logs (Application, Security, Setup and System) and also the "Applications and Services Logs" (although does not include the "Analytic and Debug" logs).
Parameters	Channel (string): The name/path of the event log to monitor. If the corresponding channel has not been enabled for event logging, the Tachyon Agent enables it automatically.

Trigger	**WindowsEventLog**
	Example: "Security" or "Microsoft-Windows-Bits-Client/Operational"
Query(string): Optional XPath expression to match against incoming entries into the event log Example: "*[System/EventID=4688]" See the MSDN references pages on Event Queries and Event XML and Event Selection for details of the supported XPath syntax and examples.	
AuditSubCategoryGuid(GUID string): Optional XPath expression to match against incoming entries into the event log. Example: "0CCE922B-69AE-11D9-BED3-505054503030" corresponds to the "Process Creation" sub-category. See the MSDN reference for a list of available GUIDs The Tachyon Agent will modify the local security policy to enable "failure" and "success" auditing for this sub-category for the period that this trigger remains active. Doing so means that corresponding event log entries will be generated when system activity occurs for this sub-category. The Tachyon Agent will attempt to re-enable this level of auditing if the Local Security Authority Subsystem Service (LSASS) indicates that the audit policy has changed - e.g. through a re-application of local/group policy, or if an administrator has manually adjusted the auditing policy using a tool like "auditpol".	
DebounceTimeSeconds(Integer): Optional de-bounce time in seconds. How long to hold back firing this trigger (to wait for additional events to occur); see the section above on Trigger de-bouncing.	

Trigger	**WindowsEventLog**
Example	Click here to expand examples...

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="AccountLogonFails" Category="Event" Description="When a logon fails due to bad username or password" Type="WindowsEventLog" ReadablePayload="When a logon fails due to bad username or password">
  <AgentParameters>
    <Parameter Name="Channel" Value="Security" />
    <Parameter Name="Query" Value="*[System/EventID=4625]" />
    <Parameter Name="AuditSubCategoryGuid" Value="0CCE9215-69AE-11D9-BED3-505054503030" />
    <Parameter Name="DebounceTimeSeconds" Value="1" />
  </AgentParameters>
</TriggerTemplate>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="AnyProcessCrash" Category="Event" Description="When any process crashes (Windows only)" Type="WindowsEventLog" ReadablePayload="On crash of any running process">
  <AgentParameters>
    <Parameter Name="Channel" Value="Application" />
    <Parameter Name="Query" Value="*[System[(Level=2) and (EventID=1000)]]" />
    <Parameter Name="DebounceTimeSeconds" Value="0" />
  </AgentParameters>
</TriggerTemplate>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<TriggerTemplate Name="FirewallEnableChange" Category="Event" Descri
ption="When a Firewall profile sta
```

Trigger	**WindowsEventLog**

```
te changes" Type="WindowsEventLog"
ReadablePayload="When a Firewall p
rofile state changes">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="Microsoft-Windows-Windows Firew
all With Advanced Security/Firewal
l" />
    <Parameter Name="Query" Value=
"*[System/EventID=2003] and *[Even
tData[Data[@Name="SettingType"] an
d (Data="1")]]" />
    <Parameter Name="AuditSubCateg
oryGuid" Value="" />
    <Parameter Name="DebounceTimeS
econds" Value="2" />
  </AgentParameters>
</TriggerTemplate>

<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="ProcessCras
h" Category="Event" Description="W
hen a process crashes (Windows onl
y)" Type="WindowsEventLog" Readabl
ePayload="On crash of process "%ex
ecutable%"">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="Application" />
    <Parameter Name="Query" Value=
"*[System[(Level=2) and (EventID=1
000)]] and *[EventData[Data='%exec
utable%']]" />
    <Parameter Name="DebounceTimeS
econds" Value="0" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"N
ame":"Executable","Pattern":"%exec
utable%","DataType":"string","Cont
rolType":"freeText","ControlMetada
ta":null,"Placeholder":"executable
with or without path","DefaultValu
e":null,"Validation":{"Regex":nul
l,"MaxLength":"1024","AllowedValue
s":null,"NumValueRestrictions":nul
l},"Value":null,"HintText":null,"S
ource":null}]]]></UserParameterJso
```

Trigger	**WindowsEventLog**

```
n>
</TriggerTemplate>

<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="ProcessLaun
ch" Category="Event" Description="
When a process starts (Windows onl
y)" Type="WindowsEventLog" Readabl
ePayload="On launch of process "%e
xecutable%"">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="Security" />
    <Parameter Name="Query" Value=
"*[System/EventID=4688] and *[Even
tData[Data[@Name="NewProcessName"]
and (Data="%executable%")]]" />
    <Parameter Name="AuditSubCateg
oryGuid" Value="0CCE922B-69AE-11D9
-BED3-505054503030" />
    <Parameter Name="DebounceTimeS
econds" Value="0" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"N
ame":"Executable","Pattern":"%exec
utable%","DataType":"string","Cont
rolType":"freeText","ControlMetada
ta":null,"Placeholder":"executable
with path","DefaultValue":null,"Va
lidation":{"Regex":null,"MaxLength
":"1024","AllowedValues":null,"Num
ValueRestrictions":null},"Value":n
ull,"HintText":null,"Source":nul
l}]]]></UserParameterJson>
</TriggerTemplate>

<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="UsbDriveIns
ertion" Category="Event" Descripti
on="When a USB drive is attached"
Type="WindowsEventLog" ReadablePay
load="When a USB drive is attached
">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="Microsoft-Windows-DriverFramewo
```

Trigger	**WindowsEventLog**

```
rks-UserMode/Operational" />
    <Parameter Name="Query" Value=
"*[System/EventID=1003] or *[Syste
m/EventID=1006]" />
    <Parameter Name="AuditSubCateg
oryGuid" Value="" />
    <Parameter Name="DebounceTimeS
econds" Value="5" />
  </AgentParameters>
</TriggerTemplate>
```

```
<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="UserAddedTo
LocalAdmin" Category="Event" Descr
iption="When an account is added a
s a local administrator" Type="Win
dowsEventLog" ReadablePayload="Whe
n an account is added as a local a
dministrator">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="Security" />
    <Parameter Name="Query" Value=
"*[System/EventID=4732] and *[Even
tData[Data[@Name="TargetUserName"]
and (Data="Administrators")]]" />
    <Parameter Name="AuditSubCateg
oryGuid" Value="0CCE9237-69AE-11D9
-BED3-505054503030" />
    <Parameter Name="DebounceTimeS
econds" Value="1" />
  </AgentParameters>
</TriggerTemplate>
```

```
<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="WindowsEven
tLog" Category="Advanced" Descript
ion="When an event log entry is cr
eated (Windows only)" Type="Window
sEventLog" ReadablePayload="On Win
dows"%channel%" event log entry ma
tching "%query%" (debounce for %de
bounce% seconds)">
  <AgentParameters>
    <Parameter Name="Channel" Valu
e="%channel%" />
```

```
    <Parameter Name="Query" Value=
"%query%" />
    <Parameter Name="DebounceTimeS
econds" Value="%debounce%" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"N
ame":"Channel","Pattern":"%channe
l%","DataType":"string","ControlTy
pe":"freeText","ControlMetadata":n
ull,"Placeholder":"channel","Defau
ltValue":null,"Validation":{"Regex
":null,"MaxLength":"1024","Allowed
Values":null,"NumValueRestrictions
":null},"Value":null,"HintText":nu
ll,"Source":null},{"Name":"Query",
"Pattern":"%query%","DataType":"st
ring","ControlType":"freeText","Co
ntrolMetadata":null,"Placeholder":
"query","DefaultValue":null,"Valid
ation":{"Regex":null,"MaxLength":"
1024","AllowedValues":null,"NumVal
ueRestrictions":null},"Value":nul
l,"HintText":null,"Source":null},
{"Name":"Debounce","Pattern":"%deb
ounce%","DataType":"string","Contr
olType":"freeText","ControlMetadat
a":null,"Placeholder":"debounce ti
me (seconds)","DefaultValue":"0","
Validation":{"Regex":"[0-9]{0,2}",
"MaxLength":"2","AllowedValues":nu
ll,"NumValueRestrictions":null},"V
alue":null,"HintText":null,"Source
":null}]]]></UserParameterJson>
</TriggerTemplate>

<?xml version="1.0" encoding="UTF-
8"?>
<TriggerTemplate Name="WindowsSecu
rityEventLog" Category="Advanced"
Description="When a security event
is created (Windows only)" Type="W
indowsEventLog" ReadablePayload="O
n Windows Security event log entry
matching "%query%" (audit category
"%auditSubcategoryGuid%", debounce
for %debounce% seconds)">
  <AgentParameters>
    <Parameter Name="Channel" Valu
```

Trigger	**WindowsEventLog**

```
e="Security" />
    <Parameter Name="Query" Value=
"%query%" />
    <Parameter Name="AuditSubCateg
oryGuid" Value="%auditSubcategoryG
uid%" />
    <Parameter Name="DebounceTimeS
econds" Value="%debounce%" />
  </AgentParameters>
  <UserParameterJson><![CDATA[[{"N
ame":"Query","Pattern":"%query%","
DataType":"string","ControlType":"
freeText","ControlMetadata":null,"
Placeholder":"query","DefaultValue
":null,"Validation":{"Regex":null,
"MaxLength":"1024","AllowedValues
":null,"NumValueRestrictions":nul
l},"Value":null,"HintText":null,"S
ource":null},{"Name":"AuditSubcate
goryGuid","Pattern":"%auditSubcate
goryGuid%","DataType":"string","Co
ntrolType":"freeText","ControlMeta
data":null,"Placeholder":"audit su
b-category GUID","DefaultValue":nu
ll,"Validation":{"Regex":"([0-9A-F
a-f]{8}[-][0-9A-Fa-f]{4}[-][0-9A-F
a-f]{4}[-][0-9A-Fa-f]{4}[-][0-9A-F
a-f]{12})","MaxLength":"36","Allow
edValues":null,"NumValueRestrictio
ns":null},"Value":null,"HintText":
null,"Source":null},{"Name":"Debou
nce","Pattern":"%debounce%","DataT
ype":"string","ControlType":"freeT
ext","ControlMetadata":null,"Place
holder":"debounce time (seconds)",
"DefaultValue":"0","Validation":{"
Regex":"[0-9]{0,2}","MaxLength":"2
","AllowedValues":null,"NumValueRe
strictions":null},"Value":null,"Hi
ntText":null,"Source":null}]]]></U
serParameterJson>
</TriggerTemplate>
```

Platforms	• Windows Vista and later.
Notes	**Warning**

Trigger	**WindowsEventLog**
	This trigger is very powerful, as many applications and operating system components use the Windows Event Log to capture activity. Additionally, the trigger can be used to monitor for Security events which depend on a specific security auditing policy.

1.7.9 WindowsRegistryChange

Trigger	**WindowsRegistryChange**
Action	Triggers when a Windows registry key changes.
Parameters	*Hive* (string): The name of the registry hive containing the key. Only *HKLM* and *HKCR* are supported, although case is insignificant.
Subkey (string): The name of the registry key.	
IncludeSubkeys (optional string, values "true" or "false", default "false"): Whether to detect changes in subkeys too.	

Example	`<?xml version="1.0" encoding="UTF-8"?>`
	`<TriggerTemplate Name="WindowsRegistryChange" Category="Event" Description="When a registry key changes (Windows only)" Type="WindowsRegistryChange" ReadablePayload="On change of registry values in "%hive%\%subkey%" (include subkeys=%includeSubkeys%)">`
	`  <AgentParameters>`
	`  <Parameter Name="Hive" Value="%hive%" />`
	`  <Parameter Name="Subkey" Value="%subkey%" />`
	`  <Parameter Name="IncludeSubkeys" Value="%includeSubkeys%" />`
	`  </AgentParameters>`
	`  <UserParameterJson><![CDATA[[{"Name":"Hive","Pattern":"%hive%","DataType":"string","Contro`

Trigger	**WindowsRegistryChange**
	lType":"valuePicker","ControlMetadata":null,"Placeholder":"hive","DefaultValue":"HKLM","Validation":{"Regex":null,"MaxLength":"4","AllowedValues":["HKLM","HKCR"],"NumValueRestrictions":null},"Value":null,"HintText":null,"Source":null},{"Name":"Subkey","Pattern":"%subkey%","DataType":"string","ControlType":"freeText","ControlMetadata":null,"Placeholder":"subkey","DefaultValue":null,"Validation":{"Regex":null,"MaxLength":"1024","AllowedValues":null,"NumValueRestrictions":null},"Value":null,"HintText":null,"Source":null},{"Name":"IncludeSubkeys","Pattern":"%includeSubkeys%","DataType":"string","ControlType":"valuePicker","ControlMetadata":null,"Placeholder":"includeSubkeys","DefaultValue":"true","Validation":{"Regex":null,"MaxLength":"4","AllowedValues":["true","false"],"NumValueRestrictions":null},"Value":null,"HintText":null,"Source":null}]]></UserParameterJson>
</TriggerTemplate>
Platforms	• Windows
Notes	Uses the Windows Registry change notification APIs to fire when a registry key changes

1.8 Custom Response Visualizations

Adding custom response visualizations to instruction definitions.

allows custom visualization of responses in the form of charts and tables in along with existing default table based response display, this is useful because:

- Users can view responses in graphical format.
- The same data can be presented from different perspectives.
- Responses can be highly summarized.

Note

This feature has been disabled from the platform 24.1.1 release.

1.8.1 Everyone likes a pretty graph

Presenting data is a tabular way is all good and well, but as the volume of data grown, it gets easier to lose the big picture. Aggregating data helps to present it in a more concise manner, but to allow the user to get a good picture of the situation in just a few seconds, you need a graph.

1.8.2 Response Visualizations

Visualizations are Tachyon's way to allow you to present the data as graphical form. In a way, they are a fancy way of saying "put this is a chart for me".

Tachyon supports several chart types: Pie, SmartBar, Line, Bar, Column, Area and StackedArea.

Pie and Area are designed as "2d" or "single series" charts while the rest are designed to be "3d" or "multi series" charts. Some of them (like Bar and Column) can be used as 2d but they don't look quite right – while the data is correct, the colours are wrong.

But the charts themselves are not enough. You need a way to process the responses so that they can be displayed in one of the charts. The processing is done by something we call a response processor. Tachyon has several processors built-in, like date-time single series, date-time multi series, default single series and default multi series. You can also provide your own code to do the processing but that is beyond the scope of this article.

1.8.3 Configuration basics

The examples below will have the raw configuration Json that each graph uses, so let's quickly go over the fields in that Json to give you an idea what they mean.

Each chart should be uniquely identified by its Id field. It is up to you what that unique value is. You can put a Title on the chart and choose from one of several chart types available.

X defines which column from the schema to use as X axis, Y does the same for value. Z axis is available on the 3d charts and allows you to pick the column over which to aggregate the data. PostProcessor defines which processing function to use and multiple charts can use the same function and essentially plot the same data differently if you so desire, or each chart can use its own processing function. Parameters of each function can be found in the documentation and they are outside the scope of this article.

Row defines which row the chart should be in. This value becomes important if you have more than one chart and helps you define if the charts should be side by side, in which case they would have the same row number, or one over another, where they would have different row numbers.

1.8.4 To aggregate or not to aggregate

Aggregated data is readily available and is relatively small in volume and can be processed quickly, even if the processor has to do extra aggregations. Dealing with raw data would mean the processor would potentially have to process hundreds of thousands of rows in order to produce the data for the graph. That would put an undesired strain on the server and should be avoided at all costs.

So, to all intents and purposes, graphs deal with aggregate data. If you want to use a visualization in one of your instructions, that instruction must have an aggregation schema and you should use columns from the aggregate schema as your X, Y and Z axis.

Please remember that using aggregated data in the chart is separate from aggregating over the Z axis in a 3D chart. 3D charts will, effectively, aggregate the input data, which has already been aggregated by the database.

1.8.5 2d vs 3d

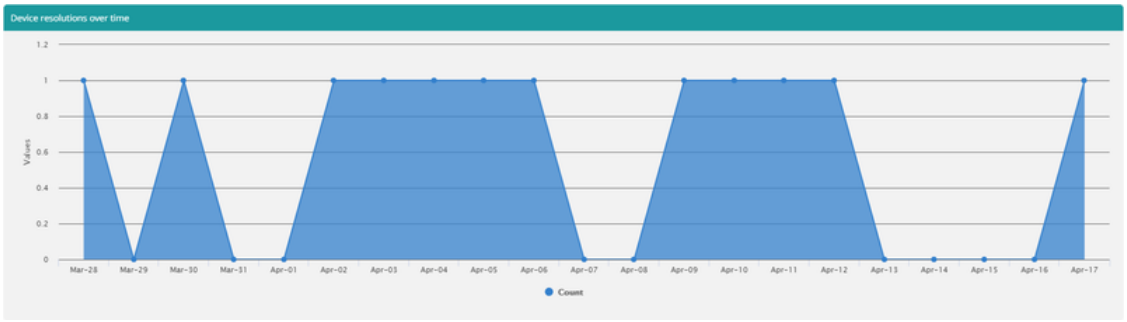
Is short, a 2d chart is a chart that displays a single series of name-value elements, while the 3d one groups entries and then displays series of name-value elements for each grouping.

This does sound a bit convoluted but in a moment, I'll show you some examples and it will become much clearer.

1.8.6 Single series (2d) charts

Let's looks at a few single series charts. First, I'll show you time-based chart that use the date time processor.

As an example we'll look at an instruction that looks on DNS resolutions over time. This instruction has a simple aggregated schema with two columns: TimeStamp and Count.



Above graph has following configuration:

JSON

```
{
	"Name": "default",
	"TemplateConfigurations": [{
		"Id": "leftArea",
		"Title": "Device resolutions over time",
		"Type": "Area",
		"X": "TimeStamp",
		"Y": "Count",
		"Z": "Fqdn",
		"PostProcessor": "processingFunction",
		"Size": 1,
		"Row": 1
	}],
	"PostProcessors": [{
		"Name": "processingFunction",
		"Function": "ProcessDateTimeSingleSeries('TimeStamp', 'Count', 'day'
, 'MMM-dd', 'Count')"
	}]
}
```

Moving away from the time-based data, we have everyone's favourite – the pie chart.

The instruction we're looking at shows count of disable services and has two columns: Caption and Count. 

And here's the configuration:

JSON

```
{
      "Name": "default",
      "TemplateConfigurations": [{
            "Id": "mainchart",
            "Title": "Count of disabled services",
            "Type": "Pie",
            "X": "Caption",
            "Y": "Count",
            "PostProcessor": "processingFunction",
            "Size": 1,
            "Row": 1
      }],
      "PostProcessors": [{
            "Name": "processingFunction",
            "Function": "ProcessSingleSeries('Caption', 'Count', '10', 'Count',
'true')"
      }]
}
```

1.8.7 Multi series (3d) charts

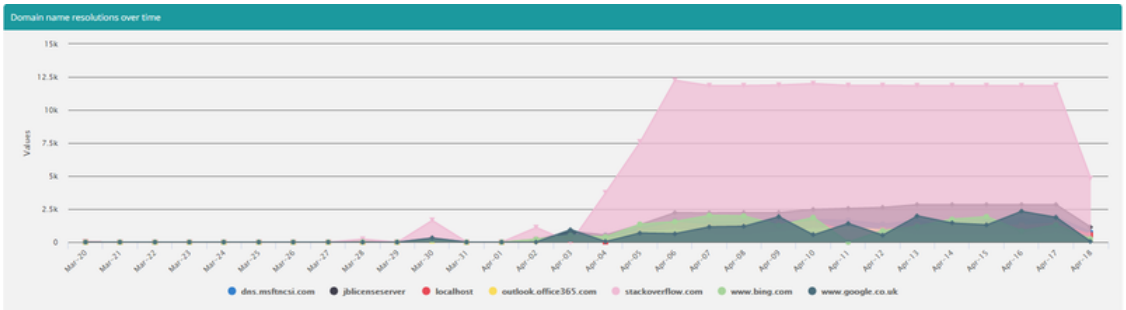
Now let's looks at a multi series chart that plots data over time

Here we have an instruction that returns aggregated data with following schema:

- Fqdn
- Timestamp
- Count

And the aggregation is performed over Fqdn and Timestamp fields. Based on that data, we'll plot two graphs.

First one shows Domain name resolutions over time.

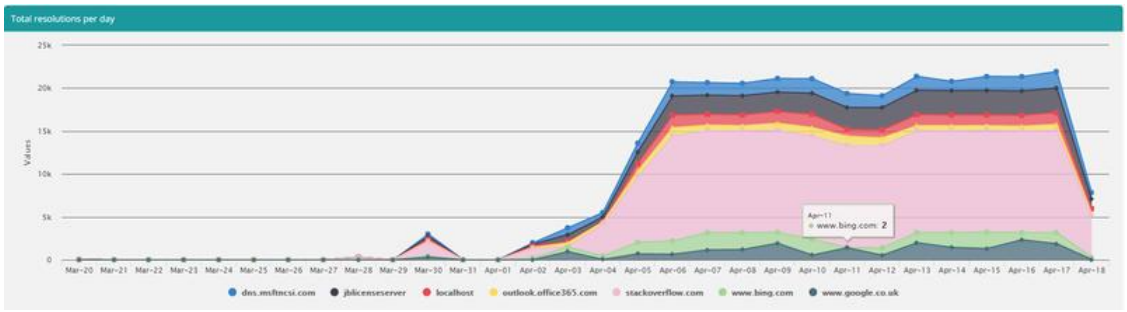


This was achieved by using following template configuration:

JSON

```
{
     "Id": "leftArea",
     "Title": "Domain name resolutions over time",
     "Type": "Area",
     "X": "TimeStamp",
     "Y": "Count",
     "Z": "Fqdn",
     "PostProcessor": "processingFunction",
     "Size": 1,
     "Row": 1
}
```

Same data is then used to plot total resolution per day:



Using this configuration:

JSON

```
{
     "Id": "bottomStackedArea",
     "Title": "Total resolutions per day",
     "Type": "StackedArea",
     "X": "TimeStamp",
     "Y": "Count",
     "Z": "Fqdn",
     "PostProcessor": "processingFunction",
     "Size": 1,
     "Row": 2
}
```

And the processor definition that serves both looks like this: **JSON**

```
{
     "Name": "processingFunction",
     "Function": "ProcessDateTimeMultiSeries('TimeStamp', 'Count', 'Fqdn', 'day
', '7', 'MMM-dd', 'false')"
}
```

As you can see we're grouping over the Fqdn column, which will give us our series. Then within each grouping we have a series consisting of a value for a given date.

The entire response visualization configuration looks like this:

JSON

```
{
     "Name": "default",
     "TemplateConfigurations": [{
           "Id": "leftArea",
           "Title": "Domain name resolutions over time",
           "Type": "Area",
           "X": "TimeStamp",
           "Y": "Count",
           "Z": "Fqdn",
          "PostProcessor": "processingFunction",
           "Size": 1,
           "Row": 1
     },
     {
           "Id": "bottomStackedArea",
           "Title": "Total resolutions per day",
           "Type": "StackedArea",
           "X": "TimeStamp",
           "Y": "Count",
           "Z": "Fqdn",
           "PostProcessor": "processingFunction",
           "Size": 1,
           "Row": 2
     }],
     "PostProcessors": [{
           "Name": "processingFunction",
           "Function": "ProcessDateTimeMultiSeries('TimeStamp', 'Count', 'Fqdn'
, 'day', '7', 'MMM-dd', 'false')"
     }]
}
```

But of course, we can plot a multi series that doesn't necessarily use time on the X axis. We can have any value there.

As an example we'll use an instruction that returns all software installed on the system. This instruction is aggregated by publisher, product and the product's version and gives a simple count. 

The chart above shows us the 5 most common apps from 5 most common publishers. To achieve that, we grouped over the Publisher and used Product as our X-axis with the count being the value. It's worth noting that this will include all versions of a given product, since taking that into account would require another level to the chart.

Here's configuration for the chart above:

JSON

```
{
     "Name": "default",
     "TemplateConfigurations": [{
           "Id": "mainchart",
           "Title": "Installed software - 5 most common application from 5 most
common publishers",
```

```
            "Type": "Bar",
            "X": "Product",
            "Y": "Count",
            "Z": "Publisher",
            "PostProcessor": "processingFunction",
            "Size": 1,
            "Row": 1
      }],
      "PostProcessors": [{
            "Name": "processingFunction",
            "Function": "ProcessMultiSeries('Product', 'Count', 'Publisher', '5'
, '5', 'false')"
      }]
}
```

1.8.8 Showing Multiple Chart types on one page

Just to wrap things up, here's an example that uses multiple graphs arranged over two rows.

Each graph has its own processor to go over data that is aggregated on Agents column and has four other columns, each representing a sum of all occurrences of a specific state (Success, Error, NotImplemented, PayloadTooLarge):

- TotalExecutionsSuccessful
- TotalExecutionsErrored
- TotalExecutionsNotImplemented
- TotalExecutionsPayloadTooLarge

And here's the full configuration Json:

JSON

```
{
      "Name": "default",
      "TemplateConfigurations": [
      {
            "Id": "TopLeftArea",
            "Title": "Agent's Operating System",
            "Type": "Pie",
            "X": "OperatingSystem",
            "Y": "Agents",
            "PostProcessor": "OsAgentCountProcessor",
            "Size": 1,
            "Row": 1
      },
      {
            "Id": "TopRightArea",
            "Title": "Successful executions per Operating System",
            "Type": "Bar",
            "X": "OperatingSystem",
```

```
            "Y": "TotalExecutionsSuccessful",
            "PostProcessor": "OsSuccessCountProcessor",
            "Size": 1,
            "Row": 1
      },
      {
            "Id": "BottomLeftArea",
            "Title": "Executions reporting an error per Operating System",
            "Type": "Bar",
            "X": "OperatingSystem",
            "Y": "TotalExecutionsErrored",
            "PostProcessor": "OsErrorCountProcessor",
            "Size": 1,
            "Row": 2
      },
      {
            "Id": "BottomRightArea",
            "Title": "Executions repoting functionality not implemented per Oper
ating System",
            "Type": "Bar",
            "X": "OperatingSystem",
            "Y": "TotalExecutionsNotImplemented",
            "PostProcessor": "OsNotImplementedCountProcessor",
            "Size": 1,
            "Row": 2
      }],
      "PostProcessors": [{
            "Name": "OsAgentCountProcessor",
            "Function": "ProcessSingleSeries('OperatingSystem', 'Agents', '10',
'AgentCount', 'true')"
      },
      {
            "Name": "OsSuccessCountProcessor",
            "Function": "ProcessSingleSeries('OperatingSystem', 'TotalExecutions
Successful', '10', 'Number of successful executions', 'true')"
      },
      {
            "Name": "OsErrorCountProcessor",
            "Function": "ProcessSingleSeries('OperatingSystem', 'TotalExecutions
Errored', '10', 'Number of executions reporting an error', 'true')"
      },
      {
            "Name": "OsNotImplementedCountProcessor",
            "Function": "ProcessSingleSeries('OperatingSystem', 'TotalExecutions
NotImplemented', '10', 'Number of executions reporting feature has not been imp
lemented on a given platform', 'true')"
      }]
}
```

And the charts themselves: 

1.8.9 Custom Response Definitions

Definition of the format of JSON object used for Custom Response Visualizations.

1.8.9.1 Response Visualization Definition

A response visualization definition must be provided in each instruction definition that requires a custom response visualization. This is because it is tightly coupled to instruction definition schema. The configuration is JSON based.

Template Configuration Properties

```
{
  "Name": "default",
  "TemplateConfigurations": [
    {
      Configuration for chart 1
    },
    {
      Configuration for chart 2
    }
  ],
  "PostProcessors": [
    {
      PostProcessor 1 definition
    },
    {
      PostProcessor 2 definition
    }
  ]
}
```

From the JSON above we can see it has three properties.

Property	Meaning
Name	This property is the name of a template. Tachyon comes with one default template named **"default"**.
TemplateConfigurations	This property contains an array of one or more template configurations. Each configuration defines how the specified template will be used, which aggregated response columns are used, and which processors are used to render the specified visualization.
PostProcessors	This property contains an array of processor definitions. Each definition specifies the configuration of a single processor. The same processor can be used by multiple visualizations.

Example

Example

```
{
  "Name": "default",
  "TemplateConfigurations": [
    {
      "Id": "column1",
      "Title": "15 most common new process starts",
      "Type": "Column",
      "X": "ExecutableName",
      "Y": "Count",
      "PostProcessor": "processingFunction",
      "Size": 1,
      "Row": 1
    }
  ],
  "PostProcessors": [
    {
      "Name": "processingFunction",
      "Function": "ProcessSingleSeries('ExecutableName', 'Count', '15', 'Count'
, 'false')"
    }
  ]
}
```

1.8.9.2 Name Property

This property is the name of a template. Tachyon comes with one default template named **"default"**. At the moment, only "default" template is available and that template handles various types of charts and graphs. In future it would be possible to create custom templates.

1.8.9.3 TemplateConfigurations Property

This property contains an array of one or more template configurations. Each configuration defines how the specified template will be used, which aggregated response columns are used, and which processors are used to render the specified visualization. At this time "default" is the only supported type of template, which provides an array of chart and graph types. Therefore, each definition describes what chart or graph can be created, which response columns are plotted, where the chart will be positioned and also how its data is to be processed using a PostProcessor.

Property	Meaning
Id	Unique chart id. It must be unique in the array of charts. **Tip**

Property	Meaning
	The id is a name, and it helps if it relates to the position on the screen (row, size) when multiple configurations (eg chart) are used within a single definition. For example, TopLeft.
Title	Title to be displayed for the chart.
Type	Supported charts are: • Pie • SmartBar • Line • Bar • Column • Area • StackedArea
X	Column name used for **label**, which must be present in aggregation schema. The label can be any datatype, except for time series processors which must be a valid date time.
Y	Column name used for **quantity**, which must be present in aggregation schema. The quantity must be a numeric dataype.
Z	Column name used for **grouping**, which if used, must be present in aggregation schema. The grouping can be any datatype. Used only for Line, Bar, Column, Area and StackedArea types.
PostProcessor	Name of the PostProcessor used to prepare responses to be consumed by the chart. The PostProcessor must be defined in the PostProcessors arrray.
Row	Row number in which the chart will be displayed. This is an integer value. One row can contain more than one chart.
Size	How much space the vizualization occupies in proportion to other vizualizations in the same row. This is an integer value. Ignored if there is only one vizualization in a row.

Template configuration example

```
{
  "Id": "TopLeft",
  "Title": "15 most common new process starts",
  "Type": "Column",
  "X": "ExecutableName",
  "Y": "Count",
  "PostProcessor": "processingFunction",
  "Size": 1,
  "Row": 1
}
```

1.8.9.4 PostProcessors Property

This property contains an array of processor definitions. Each definition specifies the configuration of a single processor. The same processor can be used by multiple visualizations. Each response processor prepares data to be consumed by a visualization and must produce data in a format expected by that visualization. V isualizations expect all the data preparation and formatting to be done by the processors and they will not do any themselves. They will simply take the processor output and attempt to use it directly as their data source. In case of charts this means attempting to plot chart specified in the visualization using the data provided by the processor.

Property	Meaning
Name	Unique name of the processor. This name is referenced in the PostProcessor field of TemplateConfigurations property.
Function	The call to a built-in processor function, used to process response data. Consists of processor name followed by a list of parameters inside parentheses. Parameters are specific to each built-in processor . Cannot be used in conjuction with Script.
Script	Custom C# code, used to process response data. Cannot be used in conjuction with Function.

Please note that Function and Script are mutually exclusive.

Built-in PostProcessor example

```
{
  "Name": "processingFunction",
  "Function": "ProcessSingleSeries('ExecutableName', 'Count', '15', 'Count', 'f
alse')"
}
```

Script based PostProcessor example

```
{
  "Name": "processingFunction",
  "Script": "var xaxis = \"ProcessName\";
          var yaxis = \"Average\";
          var yaxis = \"Average\";
          var items = responses.Select(r => { var obj = new JObject(); obj.Add(
xaxis, Convert.ToString(r.Values[\"ProcessName\"])); obj.Add(yaxis, (Convert.To
Int64(r.Values[\"Sum\"]) / Convert.ToInt64(r.Values[\"Count\"]))); return obj;
}).ToList();
          var result = new JObject();
          result.Add(\"Name\", \"Average CPU Usage (%)\");
          result.Add(\"Total\", 0);
          result.Add(\"Items\", JToken.FromObject(items));
          var cake = new List<JObject>();
          cake.Add(result);
          return JsonConvert.SerializeObject(cake);"
}
```

In the above example, the instruction that has aggregation schema with 3 columns: "Sum", "Count" and "ProcessName".

1.8.10 Built-in Processors Reference

Built-in processors for response visualizations.

Tachyon provides a number of built-in processors you can use for response visualizations. These can also be considered as inline functions.

Every processor requires X and Y axis to be specified. Multi series require Z axis as well. Each axis must be a valid field inside the 'Values' collection in your responses data.

In other words, it has to be one of your instruction's schema columns.

1.8.10.1 Function: ProcessSingleSeries

This processor can be used to create charts that plot a single series of numeric data.

Parameter index	Meaning
1	Column to use as X-axis labels. The X-axis can be any datatype.
2	Column to use as Y-axis quantities. The Y-axis must be numeric.
3	Number of entries. If this number is positive TOP entries will be selected. If this number is negative BOTTOM entries will be selected. Results will be sorted in descending order if the value of that parameter is positive and sorted in ascending order when the value is negative.

Parameter index	Meaning
4	Name of the series.
5	OPTIONAL - Flag indicating if 'other' results should be included. If set to true, anything that is not included in the number of entries defined in the third parameter will be summed together in a data point called "Others". The default value of this property is **true** (not false as you might intuitively expect). So if you don't want the additional data point you have to specify the parameter explicitly.

Example usage:

Processor call with optional parameter

```
"Function":"ProcessSingleSeries('FileName', 'Count', '9', 'Count', 'true')"
```

Processor call without optional paramter

```
 "Function":"ProcessSingleSeries('DeviceStatus', 'Count', '5', 'Count')"
```

1.8.10.2 Example: ProcessSingleSeries

The instruction we're looking at shows count of disable services and has two columns: Caption and Count. 

And here's the configuration:

JSON

```
{
     "Name": "default",
     "TemplateConfigurations": [{
           "Id": "mainchart",
           "Title": "Count of disabled services",
           "Type": "Pie",
           "X": "Caption",
           "Y": "Count",
           "PostProcessor": "processingFunction",
           "Size": 1,
           "Row": 1
     }],
     "PostProcessors": [{
           "Name": "processingFunction",
           "Function": "ProcessSingleSeries('Caption', 'Count', '10', 'Count',
'true')"
     }]
}
```

1.8.10.3 Function: ProcessMultiSeries

This processor can be used to create charts that plot multiple series of numeric data.

Parameter index	Meaning
1	Column to use as X-axis labels. The X-axis can be any datatype.
2	Column to use as Y-axis quantities. The Y-axis must be numeric.
3	Column to use as Z-axis groupings. The Z-axis can be any datatype and input data will be grouped by this column.
4	Number of entries in each series. If this number is positive TOP entries will be selected. If this number is negative BOTTOM entries will be selected. Results within each series will be sorted in descending order if the value of that parameter is positive and sorted in ascending order when the value is negative. The series themselves will be sorted in alphabetical order (ascending order).
5	Number of series. The series will be sorted in descending order if the value of that parameter is positive and sorted in ascending order when the value is negative
6	OPTIONAL - Flag indicating if 'other' results should be included. If set to true, anything that is not included in the number of entries defined in 3rd parameter will be summed together in a "Others". Also, any series outside the number of series defined in 4th parameter will be grouped together as a series called "Others".

Example usage:

Processor call with optional parameter

```
"Function":"ProcessMultiSeries('Service Name', 'Count', 'Status' '100', '10', '
true')"
```

Processor call without optional paramter

```
 "Function":"ProcessMultiSeries('DeviceStatus', 'Count', 'Status '10000', '3')"
```

1.8.10.4 Example: ProcessMultiSeries

As an example we'll use an instruction that returns all software installed on the system. This instruction is aggregated by publisher, product and the product's version and gives a simple count.



The chart above shows us the 5 most common apps from 5 most common publishers. To achieve that, we grouped over the Publisher and used Product as our X-axis with the count being the value. It's worth noting that this will include all versions of a given product, since taking that into account would require another level to the chart.

Here's configuration for the chart above:

JSON

```
{
     "Name": "default",
     "TemplateConfigurations": [{
          "Id": "mainchart",
          "Title": "Installed software - 5 most common application from 5 most
common publishers",
          "Type": "Bar",
          "X": "Product",
          "Y": "Count",
          "Z": "Publisher",
          "PostProcessor": "processingFunction",
          "Size": 1,
          "Row": 1
     }],
     "PostProcessors": [{
          "Name": "processingFunction",
          "Function": "ProcessMultiSeries('Product', 'Count', 'Publisher', '5'
, '5', 'false')"
     }]
}
```

1.8.10.5 Function: ProcessDateTimeSingleSeries

This processor can be used to create charts that plot a single series of numeric data.

This processor will establish what is the earliest and latest date time in data provided and knowing what the interval is (the unit to be used) will work out what entries are needed.

This processor expects there to be one entry per unit and will not sum entries if there happens to be more than one so it's up to you to make sure the processor is configured to handle exactly the data that your instruction produces.

This means that if you've set the interval to month and in the input data there are two entries for the same month, they will not be combined.

The series is limited to 90 elements. Older dates will be truncated.

Parameter index	Meaning
1	Column to use as X-axis labels. The X-axis must be a valid date time.
2	Column to use as Y-axis quantities. The Y-axis must be numeric.
3	Unit to be used - 'minute', 'hour', 'day', 'month', 'year' are supported.
4	Output format. Must conform to .NET DateTime data type formatting rules or custom date and time format strings.
5	Name of the series.

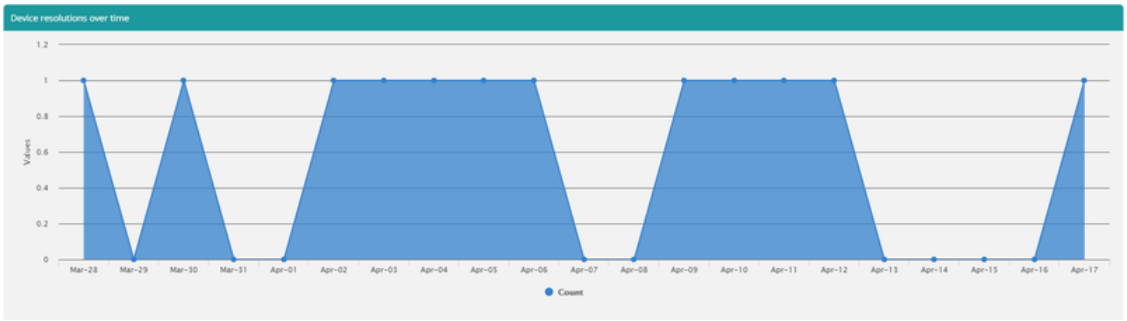
Example usage:

Processor call

```
"Function":"ProcessDateTimeSingleSeries('ModificationDateTime', 'Count', 'day', 'd', 'Modification timestamp')"
```

1.8.10.6 Example: ProcessDateTimeSingleSeries

As an example we'll look at an instruction that looks on DNS resolutions over time. This instruction has a simple aggregated schema with two columns: TimeStamp and Count.



Above graph has following configuration:

JSON

```
{
     "Name": "default",
     "TemplateConfigurations": [{
           "Id": "leftArea",
           "Title": "Device resolutions over time",
           "Type": "Area",
           "X": "TimeStamp",
           "Y": "Count",
           "Z": "Fqdn",
           "PostProcessor": "processingFunction",
```

```
            "Size": 1,
            "Row": 1
      }],
      "PostProcessors": [{
            "Name": "processingFunction",
            "Function": "ProcessDateTimeSingleSeries('TimeStamp', 'Count', 'day'
, 'MMM-dd', 'Count')"
      }]
}
```

1.8.10.7 Function: ProcessDateTimeMultiSeries

This processor can be used to create charts that plot a multiple series of numeric data.

Because there is no guarantee each series will have the same start/end date time value, this processor will use the widest time scale required by all the series and all the missing (blank) entries in each series, be that at the beginning, end or in the middle, will be filled in with zero values.

As with the single series data time processor, this processor assumes there's one entry per unit.

Each series is limited to 90 elements. Older dates will be truncated.

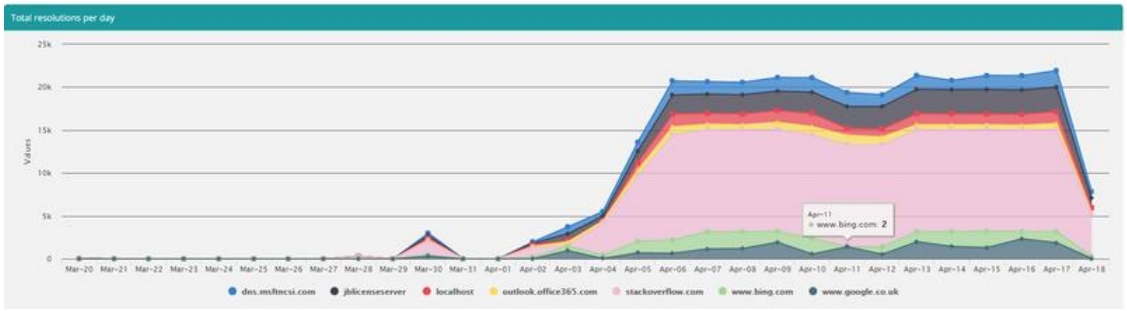
Parameter index	Meaning
1	Column to use as X-axis labels. The X-axis must be a valid date time.
2	Column to use as Y-axis quantities. The Y-axis must be numeric.
3	Column to use as Z-axis groupings. The Z-axis can be any datatype and input data will be grouped by this column.
4	Unit to be used - 'minute', 'hour', 'day', 'month', 'year' are supported.
5	Number of groups.
6	Output format. Must conform to .NET DateTime data type formatting rules or custom date and time format strings.

Example usage:

Processor call

```
"Function":"ProcessDateTimeMultiSeries('ModificationDateTime', 'Count', 'ValueT
ype' 'day', '7', 'd')"
```

1.8.10.8 Example: ProcessDateTimeMultiSeries



Using this configuration:

JSON

```
{
     "Id": "bottomStackedArea",
     "Title": "Total resolutions per day",
     "Type": "StackedArea",
     "X": "TimeStamp",
     "Y": "Count",
     "Z": "Fqdn",
     "PostProcessor": "processingFunction",
     "Size": 1,
     "Row": 2
}
```

JSON

```
{
     "Name": "processingFunction",
     "Function": "ProcessDateTimeMultiSeries('TimeStamp', 'Count', 'Fqdn', 'day
', '7', 'MMM-dd', 'false')"
}
```

1.8.11 Custom Processors

Custom processors for response visualizations.

Using the "Scripts" option in the post processor settings allows the user to inject custom C# code that will process the incoming data. These custom C# routines should be relatively short and execute quickly.

The C# you provide will be compiled on the fly and loaded into a separate Application Domain for security reasons. These domains are stored and not destroyed after use, as creating and destroying them is a very costly operation. As such, compilation is not performed every time a consumer asks for processed data.

> **Note**
>
> This feature has been disabled from the platform 24.1.1 release.

1.8.11.1 External code framework

Any code you write will be put inside a hard coded execution framework. This means that the script you write isn't a full blown C# application, but simply a body for one of the methods.

External code framework

```
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using Tachyon.Server.Data.Models;
public class Processor
{
        public static string Process(IEnumerable<Response> responses)
        {
                // your code goes here
        }
}
```

1.8.11.2 Script code

Whatever you put in the script field will be used as a body of the Process method as seen above.

This method is fed a collection of response objects. Each response, amongst other fields, has a **Values** dictionary where you can find all the columns defined in the instruction definition's schema.

Response Class (other properties omitted for brevity)

```
namespace Tachyon.Server.Common.Models
{
    using System;
    using System.Collections.Generic;
    public class Response
    {
        public Dictionary<string, object> Values { get; set; }
    }
}
```

You can use basic .NET libraries to do the processing but any resource-intensive or lengthy operations are strongly discouraged.

Your code should return a string that contains JSON that your chosen visualization uses to take the data in, as no further processing will happen. What your code return will be sent to the external consumer, which is most likely to be Javascript charting / visualization engine in Explorer.

1.8.11.3 Loaded Assemblies

Following assemblies are loaded into the script execution AppDomain

- mscorlib.dll

- System.dll
- System.Core.dll
- Microsoft.Csharp.dll
- System.Data.dll
- System.Xml.dll
- System.Data.DataSetExtensions.dll
- Newtonsoft.Json.dll
- Tachyon.Server.Common.dll
- Tachyon.Server.Data.dll

1.8.11.4 Using Types

The template class comes with a number using statements to import namespaces. If any type defined in those namespaces is used, they do not need to be addressed with fully qualified name. For example, if someone is trying to use ***Convert*** class defined in ***System*** namespace, this can be addressed just with ***Convert*** **(Convert.ToString(....))**. However, if someone wants to use a type defined in the loaded assemblies but not defined in imported namespaces they need to be referenced with a fully qualified name. For example, any reference to ***XmlTextReader*** would have to be ***System.Xml.XmlTextReader*** .

1.8.11.5 Sample processor

Below you'll find JSON configuration for a processor that uses custom processor on results gathered from an instruction that has aggregation schema with 3 columns: "Sum", "Count" and "ProcessName".

Example configuration

```
{
    "Name": "default",
    "TemplateConfigurations": [{
        "Id": "leftpie",
        "Title": "Average CPU percentage usage across devices",
        "Type": "Column",
        "X": "ProcessName",
        "Y": "Average",
        "PostProcessor": "processingFunction",
        "Size": 1,
        "Row": 1
    }],
    "PostProcessors": [{
        "Name": "processingFunction",
        "Script": "var xaxis = \"ProcessName\";
                var yaxis = \"Average\";
```

```
                var items = responses.Select(r => { var obj = new JObject(); ob
j.Add(xaxis, Convert.ToString(r.Values[\"ProcessName\"])); obj.Add(yaxis, (Conv
ert.ToInt64(r.Values[\"Sum\"]) / Convert.ToInt64(r.Values[\"Count\"]))); return
obj; }).ToList();
                var result = new JObject();
                result.Add(\"Name\", \"Average CPU Usage (%)\");
                result.Add(\"Total\", 0);
                result.Add(\"Items\", JToken.FromObject(items));
                var cake = new List<JObject>();
                cake.Add(result);
                return JsonConvert.SerializeObject(cake);"
    }]
}
```

2 Integrating with 1E

Integrate software with using public facing Application Programming Interfaces (API).

The main purpose of is to enable super-fast investigation of your network environment to retrieve sets of devices that you can then run immediate actions on.

2.1 Introduction

How you can integrate software with Tachyon using public facing Application Programming Interfaces (API).

This guide is not aimed at the general reader, it's intended primarily for use by developers who want to create their own consumers, for example using PowerShell or C#.

To get the most from this guide you should have a solid understanding of Tachyon, APIs, HTTP Communication and web API terminology.

The Tachyon API will change as new functionality is introduced and other functionality is either deprecated or removed. For this reason you should integrate with a specific version of the Tachyon Consumer API, preferably the most recent one.

The Consumer API reference shows the minimum version of the API for a given endpoint and any differences between each version.

2.1.1 What is the Consumer API?

Tachyon exposes an API called Consumer API and any software accessing this API is a Consumer. Tachyon comes with several preinstalled Consumers, like the Tachyon Explorer interface. If you access the Consumer API using an API testing tool, like Fiddler or Postman, these tools are also considered Consumers. Every Tachyon UI uses the Consumer API. There are no separate, special API endpoints for the Tachyon Explorer or any other built-in Consumer. If the built-in Consumer can do something, your Consumer will be able to do it. You'll have to create a Consumer that you'll use and make sure that it's present in your license.

Please contact support@1e.com to set up a feature in your license that will contain your Consumer.

2.1.2 Accessing the Consumer API

2.1.2.1 What am I going to need?

Any tool or programming language capable of issuing HTTP requests should be able to use the Tachyon Consumer API. This means you can use your tool of choice.

To help you use the Consumer API we have designed:

- Tachyon .NET Consumer SDK - written in .NET and providing an abstraction layer to make working with Tachyon easier
- Consumer API reference - describing available API endpoints, expected received and returned payloads, permissions
- How to guides - to perform common operations in Tachyon using the Consumer API, c overing basic use-cases with examples in raw JSON payload and Tachyon .NET Consumer SDK.

Tachyon Server versions from 3.3 have documentation you can access with a web browser. Go to **https://<MyTachyonServer>/Consumer/swagger/ui/index#/** (replace <MyTachyonServer> with the address of your Tachyon installation) to view the API documentation.

Use the the built-in swagger documentation as your primary guide regardless of your chosen tool or language and consider Consumer API Reference as back-up source.

2.1.3 How do Consumers work?

Consumers do not have to be purpose-built programs. API testing tools are also considered Consumers. A Python or PowerShell script can also be a Consumer.

Tachyon uses the concept of a Consumer to differentiate the various pieces of software accessing it. It also limits 3rd party software in its use of Tachyon (for example, by defining how many simultaneous instructions a given Consumer can issue) or define where responses to instructions should be offloaded to.

A Consumer is identified by its name and this name has to match between the license, its registration in Tachyon and each API call made. Consumer names are not case sensitive, so "MyConsumer" is considered the same as "MyCoNsUmEr". You can think of a Consumer name as an identifier, a passport if you will, that gains a piece of software access to Tachyon Consumer API. As an example, let's assume that you wish to have your own consumer and you want to call it "ScriptRunner". This means that your license file has to have a Consumer entry with the name="ScriptRunner", you have to register "ScriptRunner" as a Consumer in Tachyon and enable it, and you will have to add X-Tachyon-Consumer header field with the value of "ScriptRunner" to every API call you make using your chosen tool. Don't worry if this sounds complex, we will go through each of these elements on this page.

Many pieces of software can introduce themselves as the same consumer, for instance you can have a set of scripts using the same Consumer. Be aware that all software using the same Consumer shares that Consumer's limits and settings that are set inside Tachyon.

A Consumer must fulfill a number of prerequisites to access Tachyon. Consumers must be:

1. **Licensed** - Tachyon comes with a number of Consumers, like Explorer and Settings, that are already licensed. Any software wishing to access Tachyon must also be present in the license. The Consumer name used in the license must match the name you register with Tachyon using either Settings application or Consumer API directly.

2. **Registered with Tachyon** - a number of system Consumers are registered when installing Tachyon. All other Consumers can be registered after installing Tachyon using Settings web application or using an API call. The name of the Consumer you use here must match the name you have in your license. Note that if a piece of software wishes to register a Consumer, it doesn't have to be a Consumer itself. Permissions, will be checked as usual, read Registering a new Consumer for details.

3. **Enabled** - a setting present in the license entry for the consumer and on a Consumer registered with Tachyon.

When all three prerequisites are met, the Consumer is allowed to access Tachyon. The API call will be subject to permissions assigned using Tachyon Role Based Access Control (RBAC) to the Principal making the call.

2.1.3.1 Consumers and licensing

Consumers must be present in the license file and be enabled.

Look in your license file for an entry like the one below (where [your consumer name here] is the name of your consumer):

Sample Consumer entry in the license file

```
<Consumer name="[your consumer name here]" enable="on"> </Consumer>
```

If you cannot find this entry, please contact 1E Support and ask for an updated license file.

The feature containing your Consumer might look something like this:

Consumer feature in a license file

```
<Feature name="Customer A Consumer">
    <Consumer name="MyConsumer" enable="on"> </Consumer>
</Feature>
```

> **Note**
>
> The consumer must have enable set to equal "on".

2.1.4 Using Consumers

Each time an API call is made, Tachyon checks if the:

- Principal making the call has sufficient permissions to do that call
- Consumer they're using fulfills the three prerequisites - licensed, registered with Tachyon and enabled.

The Consumer is checked first, the Principal's permissions are checked second, only if the Consumer is successfully verified.

2.1.4.1 How does software notify Tachyon what Consumer it is?

To define what Consumer is issuing the call, add an X-Tachyon-Consumer field to the header of the HTTP call. Because each HTTP call is independent, each call must include this field in its header.

Some API endpoints do not require or check this header, but most do, so it should be included in each Consumer API call.

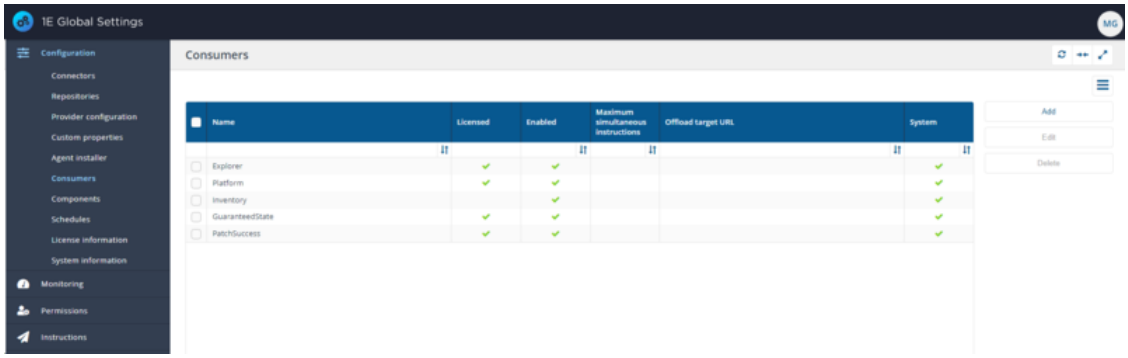
2.1.4.2 Registering a new Consumer

API endpoints relating to Consumers do not check if the software calling them is a Consumer that meets the three prerequisites listed above. This means Consumers can be added or removed by, for instance, their installation or uninstall processes. Although these API endpoints do not check for the X-Tachyon-Consumer HTTP header, they will always check the permissions of the calling Principal.

A Consumer can be registered (added) directly using the API. All the endpoints responsible for Consumer management are located in the Consumer's controller of the API. This is available under **https://<MyTachyonServer>/Consumer/Consumers** URI. Read the Consumer API reference documentation for details.

A Consumer can also be registered through the Tachyon **Settings** web application. Browse to **Configuration** and select the **Consumers** menu item.

To manage Consumers, the Principal making the call must have Write permission on Consumers' securable type. Set this by assigning the Principal to the **Consumer Administrator** role.



2.1.4.3 Consumers vs Principals

In addition to the concept of a Consumer, Tachyon has a concept of a Principals. A Principal can be a user or a group taken from Active Directory. While Tachyon does come with at least one Principal that represents a machine account (therefore a local account, not an Active Directory one), this Principal is created during installation for the purpose of internal communication between various Tachyon components and you cannot add non Active Directory Principals yourself.

A Consumer is a piece of software. A Principal is the user of that software. A "user" is something or someone using the software, for example a computer account running a service is a user even though it's not a person. Consumers are subject to Consumer check as described above. Principals are subject to RBAC checks.

2.1.4.4 How do Consumers, Principals and Permissions interact

It is important to understand how Permissions interact with Consumers and Principals.

Consumers cannot be assigned Permissions. They cannot be limited to only certain Principals either. In short, Consumers have no link or interaction whatsoever with Permissions or Principals.

Principals have Permissions through Roles they are assigned to, as per standard RBAC model.

This means that a Principal will have the same set of Permissions regardless of what Consumer they use.

If you were to launch a web browser and open Tachyon Explorer, you would have exactly the same set of Permissions as if you had opened Guaranteed State application, made an API call through Postman or run a Powershell script that contacts Consumer API.

When Princiapls and Permissions are concerned, the Consumer is irrelevant. Any Consumer can be used, as long as it is licensed and registered in Tachyon. It is only a tool that is used to perform an action. For RBAC what matter is **who** performs the action.

2.1.4.5 What happens when I make a call to Consumer API?

Tachyon performs two independent checks when it receives an API call:

1. A Consumer API check for X-Tachyon-Consumer header is performed for the vast majority of API endpoints, Consumer management and Information endpoints being notable exceptions.
    a. If given endpoint is excluded from Consumer name check, this first check concludes and the second check in performed.
    b. Consumer API will check the header of the incoming request and retrieve the value of X-Tachyon-Consumer.
        i. If this header is missing the call is rejected and an error is returned stating that X-Tachyon-Consumer field was missing in the header.
    c. Consumer API verifies that the Consumer name obtained from X-Tachyon-Consumer represents a Consumer that is known to Tachyon, that said Consumer is licensed and enabled.
        i. If any of these checks fail, the call is rejected and an error is returned that will describe the issue that caused the Consumer to be rejected, be that it not being registered, unlicensed etc.
2. Consumer API checks Permissions on the Principal making the call
    a. If the calling Principal doesn't possess the necessary Permissions to perform the operation requested, the call is rejected and an error is returned.

2.1.5 Excercise in accessing Consumer API

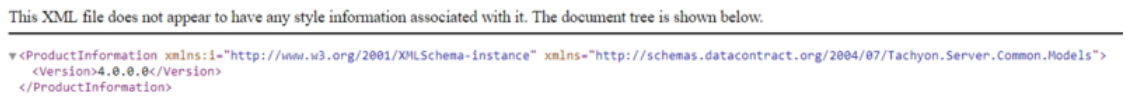
2.1.5.1 Hello world example

First we'll try to determine if you can access the API by browsing to **https://<MyTachyonServer>/Consumer/Information**.

This endpoint doesn't require the X-Tachyon-Consumer field to be present in the request's header so can be made from a web browser.

If everything works you should see a piece of XML that will state the version of the product.

Performing this step will help you establish that everything it up and running and can be accessed via given URL.



Next, access the API using an API testing tool, like Postman www.getpostman.com or Telerik Fiddler https://www.telerik.com/fiddler. With Postman (shown in the picture) remember that not all versions handle Windows Authentication correctly. Please have a look at API testing tools page which discusses several tools and their configuration.

In the examples below we'll use Postman.

First, issue a GET request to **https://<MyTachyonServer>/Consumer/SystemInformation**

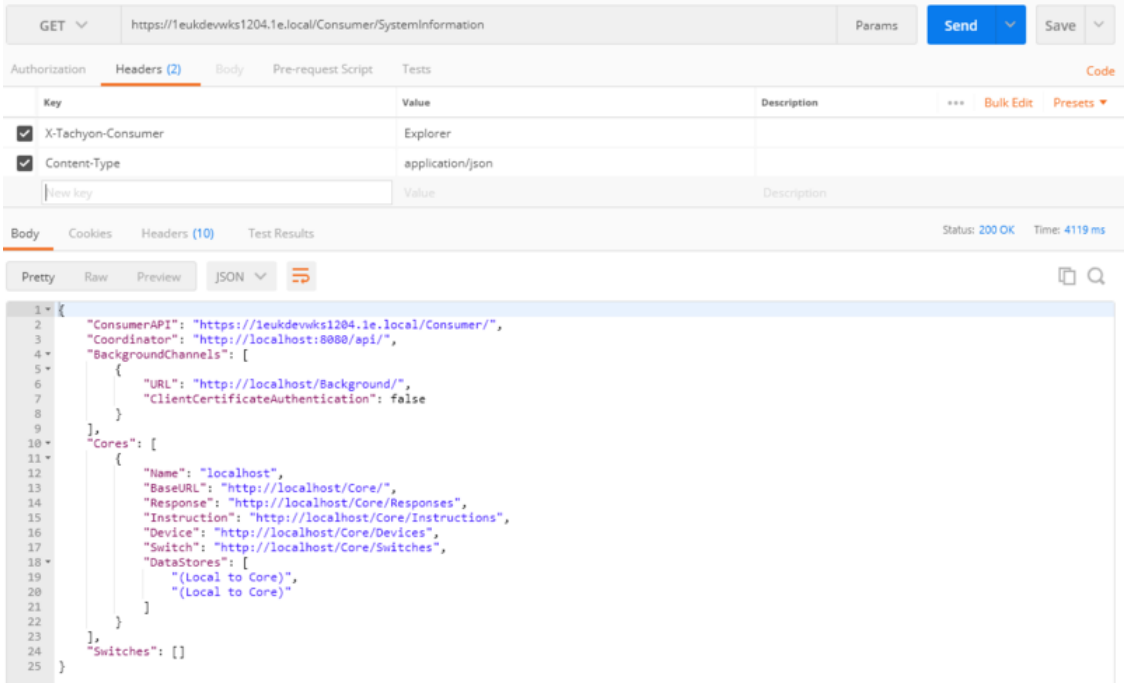
You should see something like the screenshot on the right.

This is because you didn't include **X-Tachyon-Consumer** in the request header.



Now let us add this field to the header. The example on the right is using "Explorer" as the Consumer name, but you should use the name of you Consumer.

Now when you make the request you should see something like this:



You might see something like this:

```
[
    {
        "ErrorCode": "Security.Unauthorized",
        "Message": "Unauthorized.",
        "Data": []
    }
]
```

Or like this:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>IIS 10.0 Detailed Error - 401.2 - Unauthorized</title>
        <style type="text/css">
            <!--
body{margin:0;font-size:.7em;font-family:Verdana,Arial,Helvetica,sans-serif;}
code{margin:0;color:#006600;font-size:1.1em;font-weight:bold;}
.config_source code{font-size:.8em;color:#000000;}
pre{margin:0;font-size:1.4em;word-wrap:break-word;}
ul,ol{margin:10px 0 10px 5px;}
ul.first,ol.first{margin-top:5px;}
fieldset{padding:0 15px 10px 15px;word-break:break-all;}
.summary-container fieldset{padding-bottom:5px;margin-top:4px;}
legend.no-expand-all{padding:2px 15px 4px 10px;margin:0 0 0 -12px;}
legend{color:#333333;;margin:4px 0 8px -12px;_margin-top:0px;
font-weight:bold;font-size:1em;}
a:link,a:visited{color:#007EFF;font-weight:bold;}
a:hover{text-decoration:none;}
h1{font-size:2.4em;margin:0;color:#FFF;}
h2{font-size:1.7em;margin:0;color:#CC0000;}
h3{font-size:1.4em;margin:10px 0 0 0;color:#CC0000;}
h4{font-size:1.2em;margin:10px 0 5px 0;
}#header{width:96%;margin:0 0 0 0;padding:6px 2% 6px 2%;font-family:"trebuchet MS",Verdana,sans-serif;
 color:#FFF;background-color:#5C87B2;
}#content{margin:0 0 0 2%;position:relative;}
.summary-container,.content-container{background:#FFF;width:96%;margin-top:8px;padding:10px;position:relative;}
.content-container p{margin:0 0 10px 0;
```

In the case of the first error above, you've reached Tachyon and it's telling you the Principal you're using is not authorized. This can mean that it has not been added to Tachyon or does not have permissions to use the endpoint being called. The error you see follows the Tachyon standard error reporting mechanism described in Handling errors returned by Tachyon.

If you get a web page back, like in the second error, it means your request has been rejected by IIS, not Tachyon. This can mean there's a problem with the credentials supplied in the request (incorrect username or password), your IIS settings or the tool you're using. It can also be because your tool does not support Windows Authentication, or because Windows Authentication is disabled. Tachyon uses Windows Authentication for API calls so your chosen tool must support it.

2.1.5.2 Payload format

Most API endpoints use application JSON data formatting for expected and returned data.

> **Note**
>
> Be aware that some endpoints take multi-part form-data or in a few cases return an octet stream.

Read the Swagger documentation included in the product or Consumer API reference for the details of all endpoints provided by the Consumer API.

2.1.6 Downloading the Tachyon SDK

The Tachyon .NET Consumer SDK is available from the 1E Support Portal.

1. Download the version of **TachyonSDK zip** that you require for your version of Tachyon from the .
2. Extract the files from the download zip to your choice of location.
3. Use your choice of tool or programming language, for example PowerShell or C# to create your Consumer.

For details of file versions, please refer Tachyon SDK release information.

2.1.6.1 Tachyon .NET Consumer SDK

The Tachyon .NET Consumer SDK contains an abstraction layer for the Consumer API. The layer is designed to help build Consumers by abstracting away the complexities of HTTP communication behind C# classes and providing models for the data sent to and received from Tachyon. If you're developing a .NET based application to interface with Tachyon, use the SDK library with the Consumer API reference and How to guides.

Read Tachyon Consumer SDK library for more information.

2.1.6.2 Tachyon Offloading SDK

The Tachyon Offloading SDK provides a managed DLL to use in an ASP.NET application to help create a target for Tachyon offloading. It does not access Tachyon directly, it enables the creation of software Tachyon can access during the offloading process.

Read Tachyon Offloading SDK library for more information.

2.1.7 API testing tools

The purpose of this page is to show how to configure several of the most popular API testing tools in order to make calls to Tachyon Consumer API.

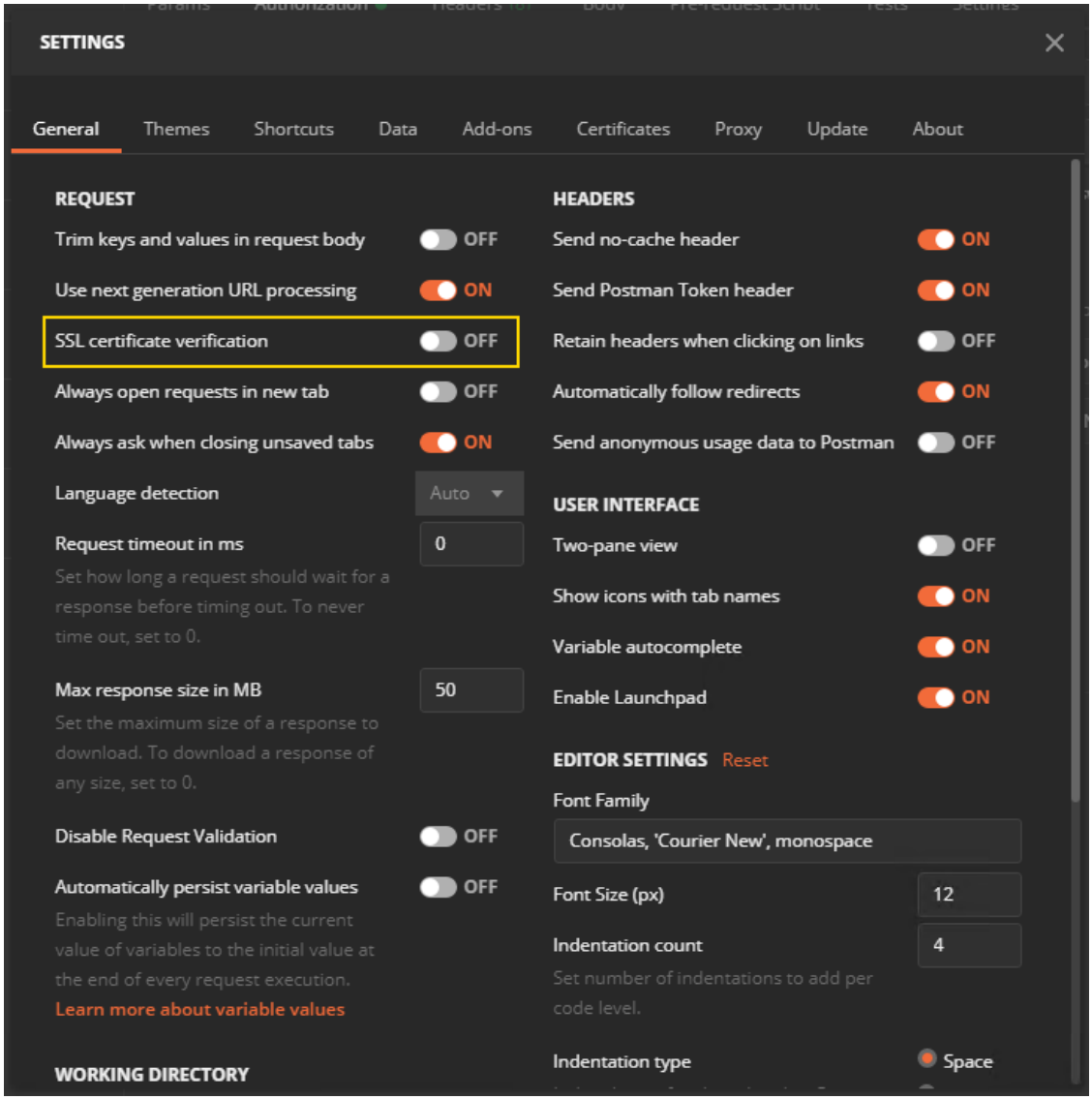
This page will not recommend a particular tool nor will it discuss differences between versions of each tool.

2.1.7.1 Postman

Version 7.36.5 was used in this guide.

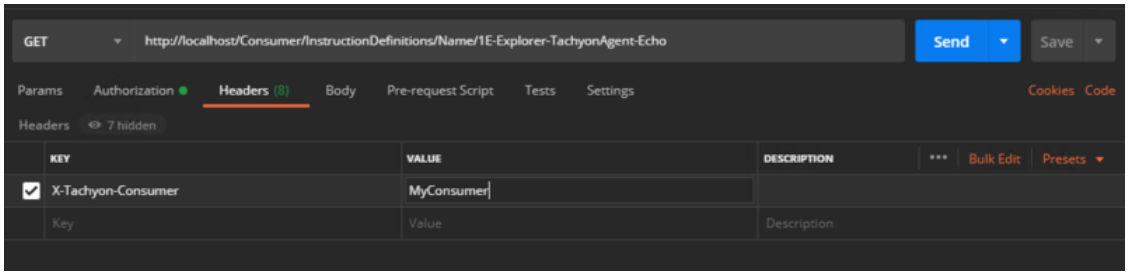
2.1.7.1.1 Turn off SSL certificate validation

Open Settings and on "General" tab turn off "SSL certificate cerification".



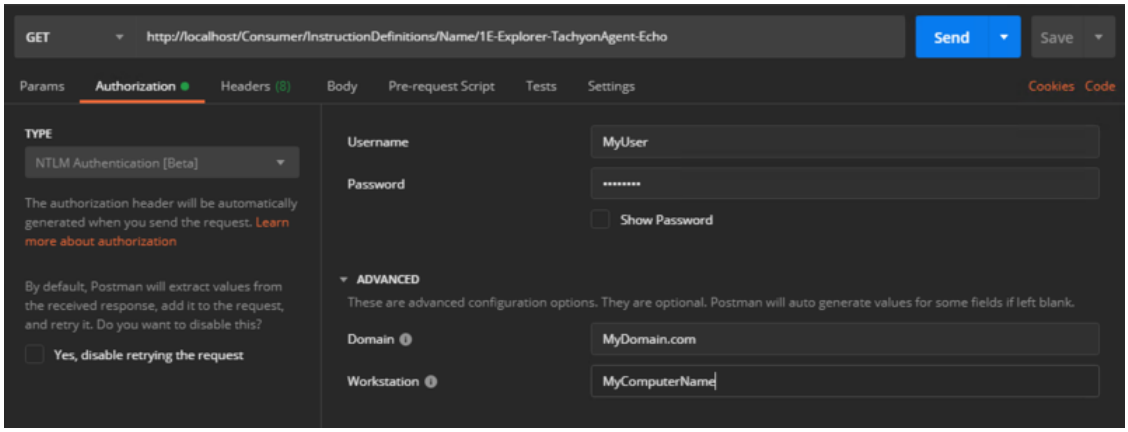
2.1.7.1.2 Setting the X-Tachyon-Consumer header field

This header fields needs to be set on the "Headers" tab ofeach request.



2.1.7.1.3 Configuring Credentials

On authorization tab, you should pick "NTLM Authentication (Beta)" and fill your credentials. You can use Username, Password and Domain or write your user name in domain\username format into the Username field.



2.1.7.1.4 Gotchas

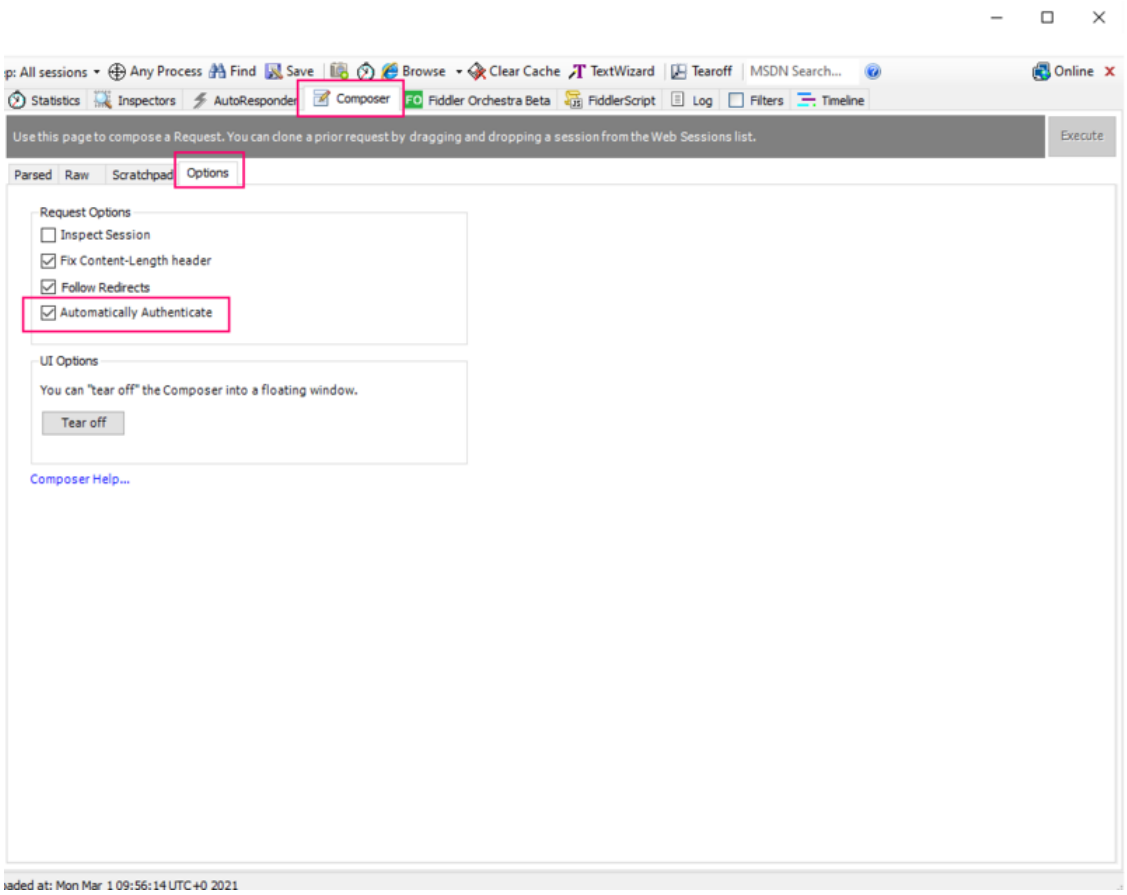
Postman caches credentials from successful API calls. This means that you can make an API calls that will succed and then make a subsequent call with different credentials and even though these new credentials should make the call fail, it succeeds. This is due to credentials caching present in Postman.

2.1.7.2 Fiddler

Version 5.0 was used in this guide.

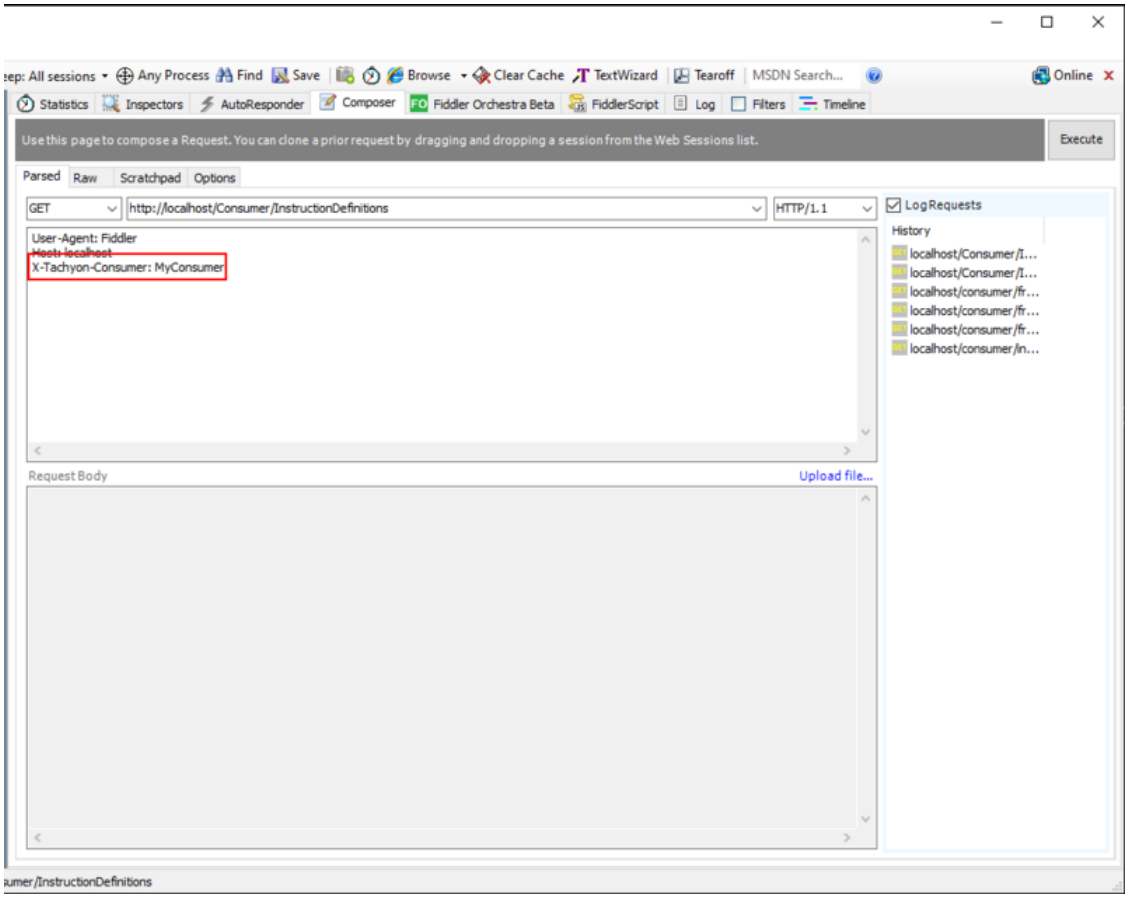
2.1.7.2.1 Enabling Windows Authentication

To turn on Windows Authentication, you should switch to "Composer", then in the "Options" tab, tick "Automatically Authenticate". This will make Fiddler use Windows Credentials of the user running it in all API calls, so there's no need to explicitly input credentials.



2.1.7.2.2 Setting the X-Tachyon-Consumer header field

Header fields can be set in "Composer" view.

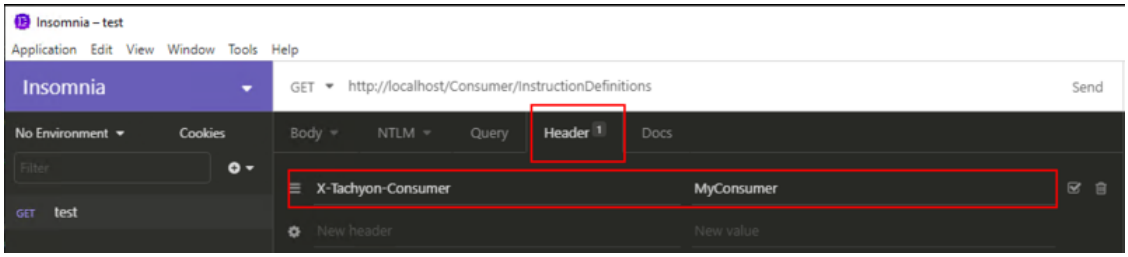


2.1.7.3 Insomnia

Version 2020.4-2 was used in this guide.

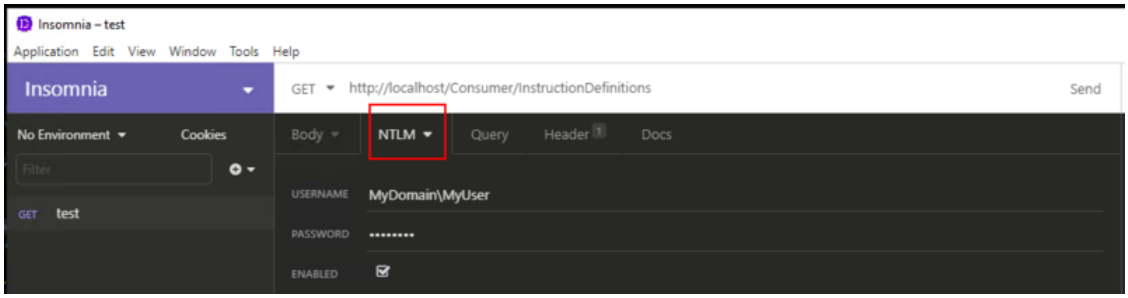
2.1.7.3.1 Setting the X-Tachyon-Consumer header field

You can set the header on the "Header" tab.



2.1.7.3.2 Configuring Credentials

You should change the authentication method to NTLM, set username in domain name \ user name format and provide a password.



2.2 Tachyon RunInstruction command-line tool

Important

We have deprecated Tachyon.RunInstruction.exe in 1E platform v9.1.

The Authentication Proxy requirement has been removed from the 1E Toolkit, which now works as is, and can be directly accessed using the 1E platform URL.

You can also use the #UUIDaf64567be51c06a05fbdac221f31188d to run instructions in addition to Endpoint Troubleshooting.

Tachyon RunInstruction command-line tool is a stand-alone command line (CLI) tool that supports submission of an instruction, question or action to a set of computers.

RunInstruction does not track an instruction once it's submitted, so you'll need to use the Endpoint Troubleshooting or other tools to track an instruction's progress and results.

Read Installation and configuration for how to install the 1E Toolkit and register a consumer.

2.2.1 Running an Instruction

Use RunInstruction with the following parameters:

```
Tachyon.RunInstruction –I instruction_name [–P param1 param2 …] –C consumer_API
_URL [-U username password] [-V] [-N ConsumerName] [-O] -D device1 [device2 …]
| -F devices_file
```

If a parameter contains spaces, wrap it in double-quotes.

Option letters are not case-sensitive, and forward slashes can be used instead of hyphens for the options if desired. The options (followed by their associated values) can be in any order, but cannot be repeated.

2.2.1.1 RunInstruction parameters

Parameter	Description
-I instruction_name	Is the name of the Tachyon instruction to run, as defined in its product pack. The product pack must have been imported into Tachyon. Some instructions consist of a single word, others contain a number of space-separated words (where the value should be wrapped in quotes).

Parameter	Description
-C consumer_API_URL	Is the URL of the Tachyon Consumer API in the form *https://server/Consumer* or (from Tachyon v2.1) just *https://server* (although the */Consumer* suffix can still be given).
-P param1 param2 ...	A list of the parameters (if any) the instruction requires. Some instructions do not take parameters. Read Instruction Parameters for details.
-U username	An optional account username to access the Tachyon Consumer API. If it's not specified the account of the user running the tool is used.
password	An optional account password to access the Tachyon Consumer API. **Note** Give both a username and password or give neither.
-V	An option specifying that device names should be verified against DNS and giving a warning for every device unrecognized by DNS. This is optional because it can be slow.
-N ConsumerName	The name that's passed to the server in the X-Tachyon-Consumer HTTP header. The default is "TachyonRunInstruction". Any custom consumer name must be prefixed with "TachyonRunInstruction".
-O	Specifies that results of the instruction should be Offloaded according to the offloading configuration set for this Consumer on the Tachyon server.
-D device1 device2 ...	A list of one or more devices identified by their FQDNs (fully-qualified domain names). The (single) Tachyon instruction will only be sent to these devices.
-F devices_file	A file containing the list of devices identified by their FQDNs. The file can be either a text file with devices separated by whitespace (for example split per line or space- or tab-separated) or a *.csv* file with an *FQDN* column (case is not significant). Duplicates are filtered out. The (single) Tachyon instruction is only sent to these devices. This is an alternative to listing the devices on the command line.

2.2.1.2 Instruction parameters

All instruction parameters consist of a name and a value. Name matching is case-insensitive, but the case of each value is preserved. Every parameter name defined in the instruction's manifest in the *<ParamaterJson>* section must be present with a value. Missing parameters and extra parameters are treated as an error. This tool does not validate the parameter value (for example, in terms of type or a pattern that can be defined in the manifest), only that a required value is present.

Each parameter has the format: *name:value*

The value can contain colon characters – The first colon marks the end of the *name* part. The value can be empty, for example *tagvalue* specifies an empty string value.

2.2.2 Examples

Note

If you copy and paste the following examples, beware of hyphen characters getting lost. You may paste what appears to be a hyphen, but is not.

2.2.2.1 Simple example

This example involves an instruction that takes no parameters.

```
Tachyon.RunInstruction.exe -I 1E-Explorer-TachyonCore-InstalledMemoryDetails -C
https://tachyon.inv02.local/Consumer -V -D inv02-cm01.inv02.local inv02-cm02.inv0
2.local
```

This submits the 1E-Explorer-TachyonCore-InstalledMemoryDetails instruction which gets details of RAM chips. The Tachyon Consumer API server is *tachyon.inv02.local*. The 2 devices on which the instruction runs are *inv02-cm01.inv02.local* and *inv02-cm02.inv02.local* and there is a check that those FQDNs are known hosts. This instruction takes no parameters.

If submission is successful - the resulting instruction ID is displayed on the CLI and logged. The ID can then be used to monitor the progress of the instruction using the Tachyon Explorer.

If submission fails - (for example, the instruction is not known to Tachyon because its product pack has not been imported, or the Consumer API URL is not correct), error messages are displayed and logged.

2.2.2.2 Example of Instruction parameters

```
Tachyon.RunInstruction.exe -I "1E-Explorer-TachyonCore-RegistryEnumerateValues"
-P "hive:HKLM" "subkey:Software\1E\Client\Persist" -C https://tachyon.inv02.loca
l/Consumer -U myusername  mypassword -D inv02-cm01.inv02.local inv02-cm02.inv02.l
ocal
```

This submits the 1E-Explorer-TachyonCore-RegistryEnumerateValues instruction (from the 1E-Explorer-TachyonCore product pack, found under Classic product packs). As in the example above, the Tachyon Consumer API server is *tachyon.inv02.local*. The 2 devices on which the instruction is run are *inv02-cm01.inv02.local* and *inv02-cm02.inv02.local*.

In this case there's no check that those FQDNs are known hosts (because there is no *–V* option). The name and password of an account with access rights to submit an instruction have been specified. If this question is successful, the values under *HKLM\Software\1E\Client\Persist* in the registry of the specified devices are retrieved.

2.2.2.3 Example: offloading the results of an instruction

```
Tachyon.RunInstruction.exe -I 1E-Explorer-TachyonCore-BiosDetails -C http://loc
alhost/Consumer -N SyslogRelay -O -D client1.acme.local
```

This submits the "1E-Explorer-TachyonCore-BiosDetails" instruction. The Tachyon Consumer API server is *Localhost*, which is the command-line tool from the same computer where Tachyon is installed. The -N parameter specifies the the Consumer Name *SyslogRelay* should be used. This consumer has to be previously configured from the Administration pages in Tachyon. Finally, -O indicates the results should be offloaded (to whatever address was configured for the consumer) instead of saved to the Tachyon Responses database.

2.2.3 Installation and configuration

The **RunInstruction** tool is installed as part of the 1E Tachyon Toolkit, along with the optional Configuration Manager Console extensions. Please refer to Extending Tachyon documentation for the relevant version of your Tachyon Platform.

The default installation location is:

%ProgramFiles(x86)%\1E\Tachyon\Toolkit\TachyonRunInstruction.

You can then copy the TachyonRunInstruction folder to be used elsewhere.

2.2.3.1 Prerequisites

RunInstruction installation prerequisites:

Tachyon installed with its Consumer API web service - for example:
https://tachyon.inv02.local/Consumer

In Tachyon make sure you have:

-
  - Tachyon user(s) with a known username and password
  - The *TachyonRunInstruction* consumer registered in Tachyon
  - Imported relevant Tachyon instructions into Tachyon
  - Permissioned imported instructions for the Tachyon user(s)
  - Permissioned for at least one Managament Group, for example **All Devices**.
  - Uploaded into Tachyon the instructions you wish to issue.

On the remote client you have:

-

- .NET Framework 4.5 or later
- A copy of the Tachyon Toolkit.

2.2.3.2 Registering a consumer

Register a consumer with **Name** *Tachyon RunInstruction* and **Enable** it. Its other properties can be left at their defaults.

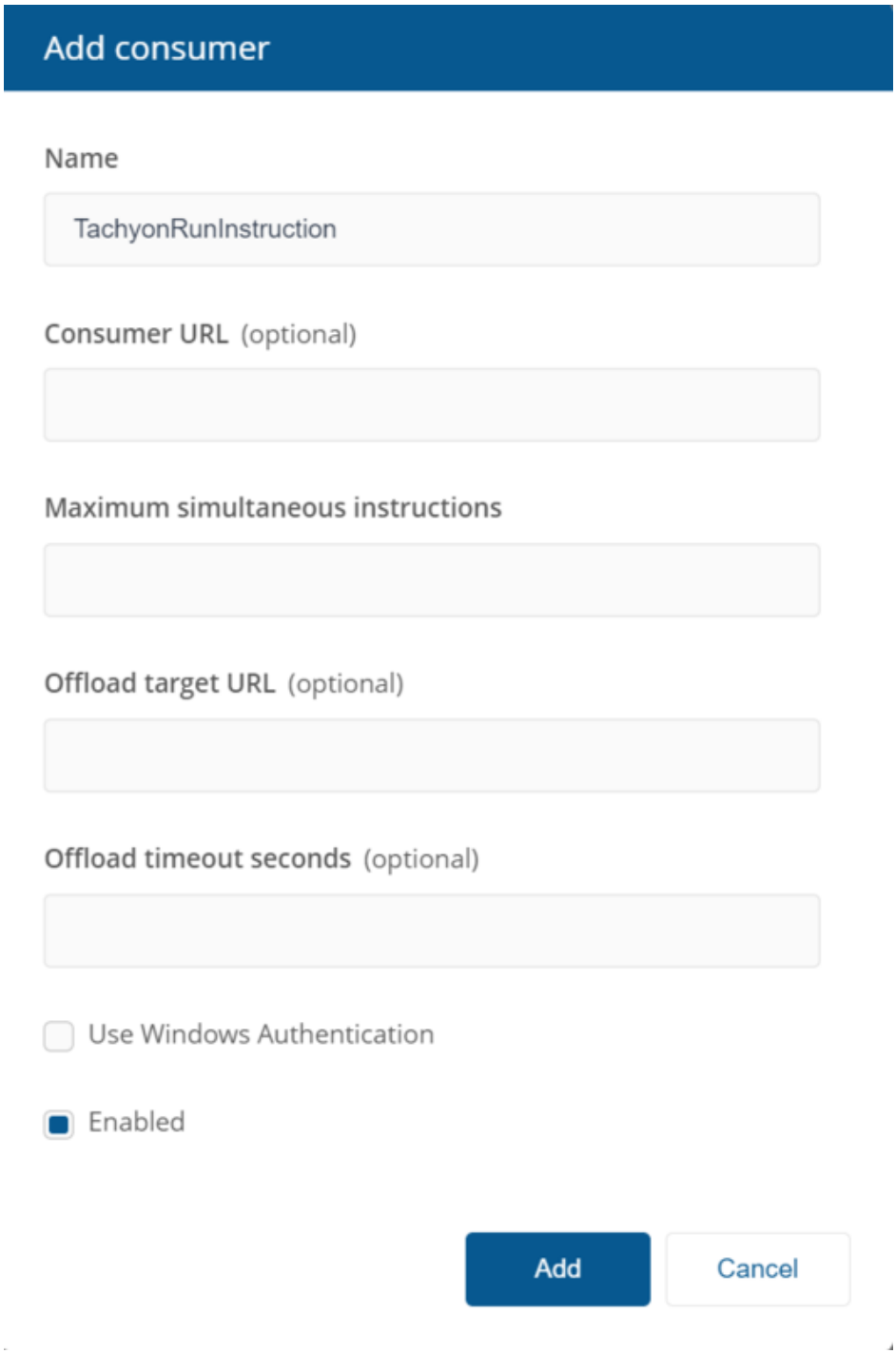
Note

If you use the **-N** option to specify a different consumer name at runtime, register and enable a different name prefixed with *Tachyon RunInstruction*.

If you use the **-O** option, specify the **Offload target URL**. For example https://tachyon.inv02.local/OffloadingClient/api/Responses

You can register multiple Consumers with variations of names and offloaders.

2.2.3.2.1 Go to the Tachyon Portal and navigate to Settings→Configuration→Consumers



2.2.3.3 Files

The Tachyon Toolkit installer places these files in *%ProgramFiles(x86)%\1E\Tachyon\Toolkit\TachyonRunInstruction*.

File	Description & Notes
log4net.dll	Provides logging.
Newtonsoft.Json.dll	JSON serialize deserialize library used to communicate with the Tachyon Consumer API.
System.Data.SQLite.dll	SQLite library.
Tachyon.ConsumerApi.Interface.dll	Library to access the Tachyon Consumer API.
Tachyon.RunInstruction.exe	The tool itself.
Tachyon.RunInstruction.exe.config	Configuration file for the tool, controlling logging etc.
Tachyon.SDK.Consumer.dll	Consumer API SDK library.

2.2.3.4 Tool configuration

Configuration is controlled by *Tachyon.RunInstruction.exe.config*- see above for its location. It controls logging and output, and should not need modification. Any output is sent both to the console and with more details to the logfile.

The default logfile path is:

C:\ProgramData\1E\Tachyon\Tachyon.RunInstruction.log

The default logging level is *INFO* (which also displays errors and warnings). Read http://logging.apache.org/log4net/release/manual/configuration.html for more details about log configuration.

2.2.4 Diagnostics

If the command fails with an error of the form *Consumer API returned BadRequest (Bad Request)* it may not have been registered as a Tachyon consumer (see Prerequisites).

2.3 Consumer API reference

Each API endpoint is documented on a separate page. Pages are version controlled.

2.3.1 Overview

This page lists every endpoint provided by the Consumer API which is a RESTful API consumed by the UI and other Consumers. Although has other APIs, the Consumer API is the only one supported for use by third party Consumers.

The URL used for requests is https://{tachyonConsumerServer}/Consumer/{request}

The text shown in example responses is spaced out for readability, normally they are compacted JSON on a single line with no white space between fields to minimize transmitted data.

API endpoints can be viewed and tested using a tool like the Postman, Fiddler or Advanced REST client extension for Chrome, but your chosen tool must support Windows Authentication.

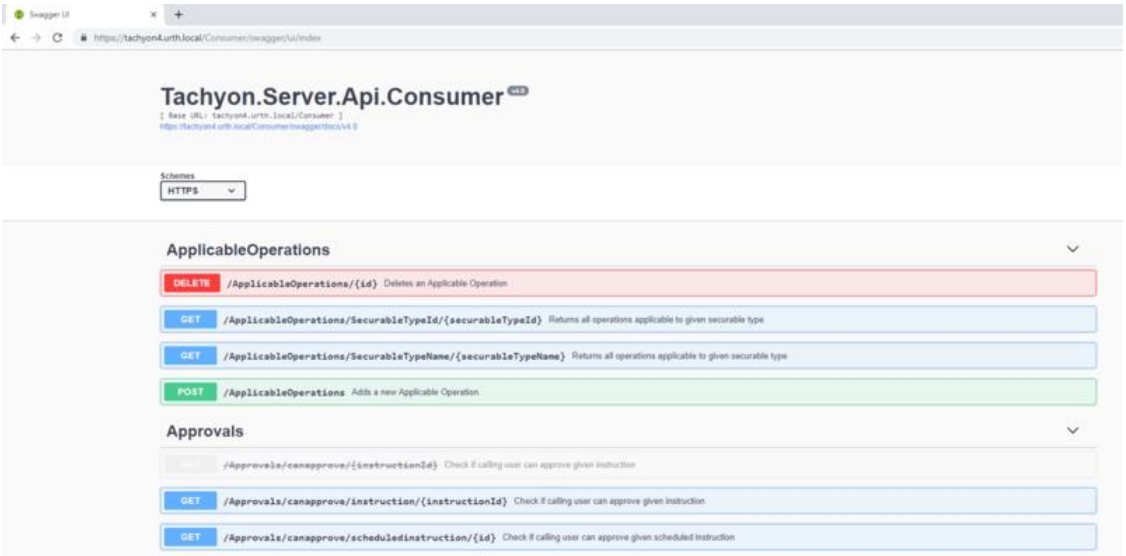
> **Note**
>
> In lab and test environments it's possible to configure to use Basic Authentication over HTTPS using tools that only support this protocol where you would pass credentials explicitly rather than rely on NTLM challenge/response authentication.
>
> This is useful where the account context in which the API call is made is not the account context in which you'd like the call processed. For example, when interacting with subsystems such as Workflow you may wish to make calls in the context of the requester and the responder but without having to switch interactive desktop sessions as the caller.

2.3.2 Exploring the API using a web browser

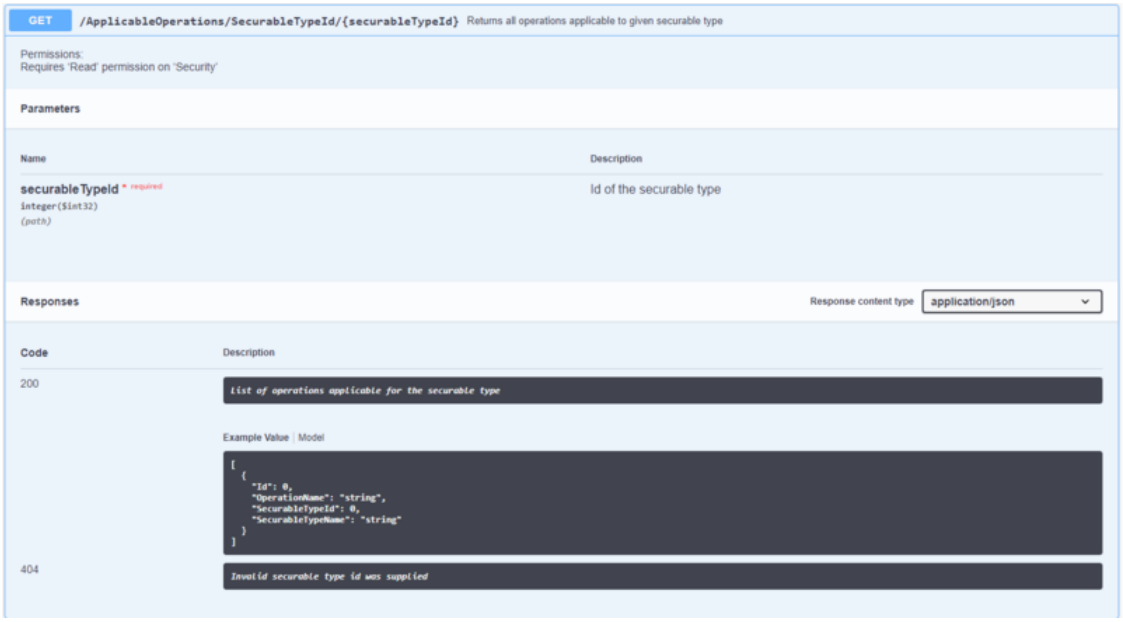
Although the Consumer API is documented in the child pages to this page, you can also explore it using the inbuilt API documentation. This is available through your browser and managed through the Swagger subsystem by browsing to the Consumer reference.

`https://<tachyon server>/Consumer/swagger`

Where <tachyon server> is the URL of your Tachyon server, which redirects to the API reference page:



Each API is expandable, by clicking on it, you can view additional information about the API.



2.3.2.1 Swagger URL reference for platform

Where <tachyon server> is the URL of your server, which redirects to the API reference page.

Component	URL
Portal	`https://<tachyon server>/Tachyon/swagger/ui/index`
Consumer	`https://<tachyon server>/consumer/swagger/ui/index`
	`https://<tachyon server>/experience/swagger/ui/index`
SLA Admin	`https://<tachyon server>/admin/swagger/ui/index`
SLA Platform	`https://<tachyon server>/platform/swagger/ui/index`
	`https://<tachyon server>/ContentDistribution/docs/index.html`

Note

Content Distribution documentation is not readily available, please follow these steps to enable it.

1. Locate *%ProgramFiles%\1E\ContentDistribution\Web\WebService\appsettings.json*

Component	URL

2. Add the **AllowSwagger** key in the **AppSettings** section. The settings should look like **"AllowSwagger": true**
3. Reset the IIS.

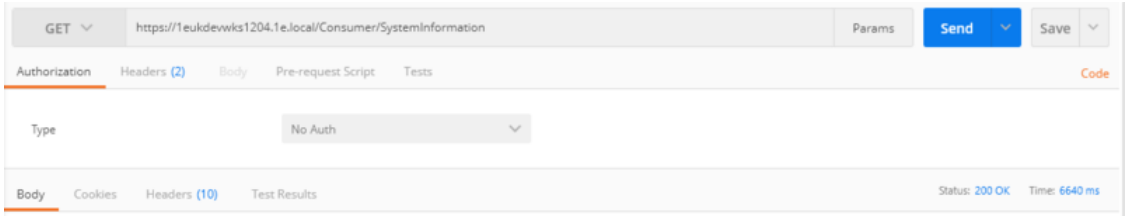
2.3.2.2 Tool settings

Authentication Mechanisms should be read in conjunction with this page to understand the different options available

Postman

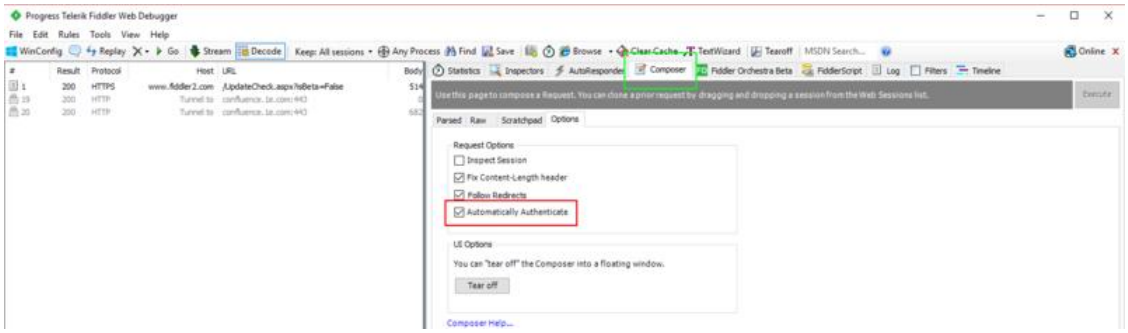
In Postman this setting and the Authentication, varies from version to version. For example, in the Chrome App version it's done automatically when **No Auth** option is selected.

In new standalone application, you should select the relevant type of Authentication



Fiddler

In Fiddler check **Automatically Authenticate** on the **Composer->Options** tab.



Advanced REST client

Google App Runtime for Chrome (ARC) requires the XHR extension turned on (the triple dot menu, top right of the window).

2.3.2.2.1 Authentication Mechanisms - Introduction

This page covers authentication methods supported by the platform, focusing on the differences between “legacy” Integrated Windows Authentication mode and “modern” token-based system that integrates with OAuth.

2.3.2.2.1.1 Types of Authentication Available

If you're using 8.1, only Integrated Windows Authentication is available and your IIS must have Windows Authentication enabled and Anonymous Authentication disabled.

If you're using version greater than 8.1, only token-based authentication is available and your IIS must have Windows Authentication disabled and Anonymous Authentication enabled.

Whereas version 24.1 and later only supports token based authentication and have no IIS as these are cloud native.

Which specific identity provider is supported depends on particular version of . For instance, in 8.2 Azure AD is supported while 8.3 adds support for Okta. Please consult documentation specific to your version of for a complete list and information on how to configure each Identity Provider (IdP).

The Windows Active Directory is supported in 1E version 9.1 and follows token based authentication, where Windows AD is wrapped in a special layer that makes it appear as if it is an OAuth Identity provider.

2.3.2.2.1.1.1 Integrated Windows Authentication

This was the original authentication method. It is based on Windows accounts, either local or domain, where domain accounts would be used by users and groups while local accounts would be used by services.

2.3.2.2.1.1.1.1 Integrated Windows Authentication - The Basics

The burden of handling authentication rests on Windows itself. A REST call via a tool or a browser would have to contain appropriate Windows credentials and IIS would require those via a 401 negotiation.

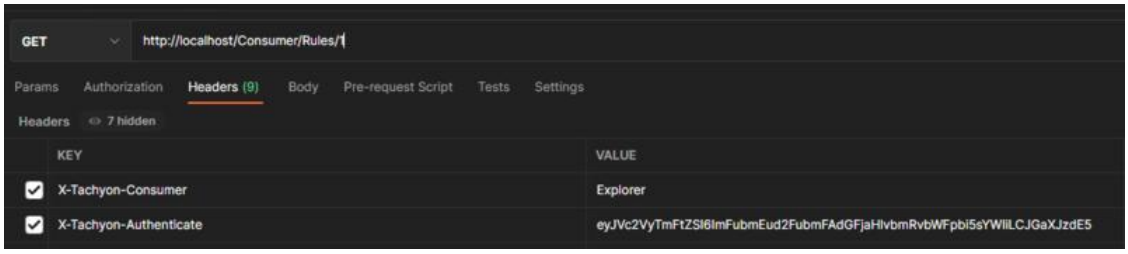
In most cases, authentication is automatic because a user is logged into a domain and therefore their credentials are available and can be supplied without user inputing them in. In some circumstances (such as running the browser in incognito mode) the browser would display a popup requesting credentials from the user, but in most cases authentication is invisibly handled by IIS and the browser.

2.3.2.2.1.1.1.2 Token based authentication and OAuth support - Contacting the API

Once you have a Token you can call APIs.

To do so, set your authentication to "no authentication" and add a header field called "X-Tachyon-Authenticate" into your request. The value of this field should be the Token. Do not add any extra characters, quotes or anything else like that. The value of this field is expected to be just the token.

Here's an example of how this would look in Postman:



2.3.2.2.1.1.2 Token based authentication and OAuth support

Token based authentication is abstracted from any particular authentication system and completely independent from Windows.

2.3.2.2.1.1.2.1 Token based authentication and OAuth support - The basics

This system will rely on an external IdP provider (usually an OAuth compliant one like AzureAD or Okta) to handle authentication process as far as verifying user names and passwords in concerned.

This system will not, however, rely on tokens issued by said IdP as an authentication method inside . Instead, will issue its own cryptographicaly signed tokens based on the IdP's tokens.

This means that a token needs to be requested and logging in is no longer an automatic affair as it was with Integrated Windows Authentication.

Instead, the user must begin their interaction with by requesting authentication. A response to this request will be a web address that points to the configured IdP. There the user will input their credentials and log into their IdP (be that AzureAD, Okta or another provider) as they would for any other website. Afterwards, if the authentication was successful, the IdP will call back to notify it of successful authentication. At that point will obtain IdP's token, keep it and exchange it for one of it's own tokens that it will give to the caller who requested authentication.

So, the end result of an authentication request should be a token.

2.3.2.2.1.1.2.2 Token based authentication and OAuth support - How the system works

Contacting the API requires a Token.

In order to obtain a Token, you have to complete an authentication request through the platform. This will entail requesting authentication from the platform and authenticating with your IdP. Once that is complete, you will receive a Token.

You can perform interactive authentication via your IdP's UI or use a programmatic way like JWT (which requires extra setup).

From that point onwards, you should add that token to every API request you make. Not just to Consumer API, but every API in the platform.

2.3.2.2.1.1.2.3 Token based authentication and OAuth support - Obtaining a taken

2.3.2.2.1.1.2.3.1 Token based authentication and OAuth support - Interactive login

For 1E Platform version 8.2-

In order to obtain a Token you have to make a GET request to wss://your.tachyon.server/Tachyon/api/Authentication/RequestAuthentication.

This request must be made using websocket protocol in order to maintain constant connection throughout the entire authentication process. Should the connection be interrupted for whatever reason before the authentication has finished, you will have to start again. Resuming authentication or polling aren't supported.

Once the above request is made, you will receive a redirect HTTP code and a URL address to follow. This address should be used to authenticate with your configured IdP. Once that is successful and the IdP has called back (please note you will not be aware of this taking place as it will be the IdP calling API directly) you will receive your Token through the websocket connection. At this point the authentication request is considered complete and websocket connection can be closed.

For 1E Platform version 9.0 and above-

In order to obtain a Token you have to make a GET request to https://your.platform.server/Tachyon/api/Authentication/RequestAuthentication (for 9.X and 23.X) or https://your.platform.server/api/Authentication/RequestAuthentication for versions 24.1 newer.

This is a regular GET and unlike 8.2 it does not require a websocket protocol to be used.

Once the above request is made, you will receive a redirect HTTP code and a URL address to follow. This address should be used to authenticate with your configured IdP. Once that is successful and the IdP has called 1E Platform back (please note you will not be aware of this taking place as it will be the IdP calling 1E Platform's API directly). You will have to pool to check the status of your request to find out when you can retrieve the token. For more details on how to implement this method see Implementing interactive authentication.

2.3.2.2.1.1.2.3.2 Token based authentication and OAuth support - Non-interactive Login

You can also obtain a Token using a JWT you already have.

In order to do so you should make a POST request to https://your.tachyon.server/Tachyon/api/Authentication/RequestJwtAuthentication (for 9.X and 23.X) or https://your.platform.server/api/Authentication/RequestJwtAuthentication (for versions 24.1 and newer) and provide your token in the body of the request. This endpoint expects a string so your token needs to be wrapped in double quotes in the payload. If the JWT is valid you will receive a Token in return.

2.3.2.2.1.1.2.3.3 Token based authentication and OAuth support - The toolkit

The toolkit has the ability to retrieve a Token.

Please consult toolkit's documentation for details: 1E PowerShell Toolkit management cmdlet reference.

2.3.2.2.1.1.2.4 Token based authentication and OAuth support - Handling Tokens

2.3.2.2.1.1.2.4.1 Token based authentication and OAuth support - Token types

There are two main types of Token, an interactive token that represents a user or a group, and a non-interactive token that represents an element of platform, an API, a service or another piece of software.

Token based authentication and OAuth support - Token Types - Interactive

This is the token you will be obtaining sing the authentication mechanisms (both interactive and JWT) described above. This type of token has two subtypes.

First subtype is the fully interactive type, obtained via interactive login as described above. These tokens can be refreshed (see below) up to a number of times defined in your configuration.

Second subtype is the semi-interactive, obtained by supplying a JWT. These tokens cannot be refreshed. You need to obtain a new JWT and supply it to the API to obtain a fresh Token.

Beyond the refresh, the two token subtypes are functionally identical.

Token based authentication and OAuth support - Token Types - Non-interactive

This type of token is used by internally. You will not be able to obtain this type of token.

2.3.2.2.1.1.2.4.2 Token based authentication and OAuth support - Token lifecycle

Fully interactive tokens can be refreshed. This is done by making a POST request to https://your.tachyon.server/Tachyon/api/Authentication/RefreshToken (for 9.X and 23.X) or https://your.platform.server/api/Authentication/RefreshToken (for versions 24.1 newer) and providing the token you'd like to refresh in the body of the request. This endpoint expects a string so your token needs to be wrapped in double quotes in the payload. Any and all API calls should check error returned to detect CommonAuthentication.TokenExpired error, which indicates that the token has expired and needs to be refreshed.

Tokens can only be refreshed a certain number of times before you will be forced to perform the authentication from scratch. If you attempt to refresh a token too many times, an error will be returned by the token refresh API when that happens and the error code will be CommonAuthentication.SessionReachedMaximumNumberOfRefreshes.

You can log out of the system by calling https://your.tachyon.server/Tachyon/api/Authentication/Logout (for 9.X and 23.X) or https://your.platform.server/api/Authentication/Logout (for versions 24.1 newer). This is a GET request that does not need any parameters, but it needs to have a token in the header field (see below). This will remove the ability of the token to be refreshed, effectively ending the "session", though for technical reasons your token will remain valid for the remainder of its lifetime.

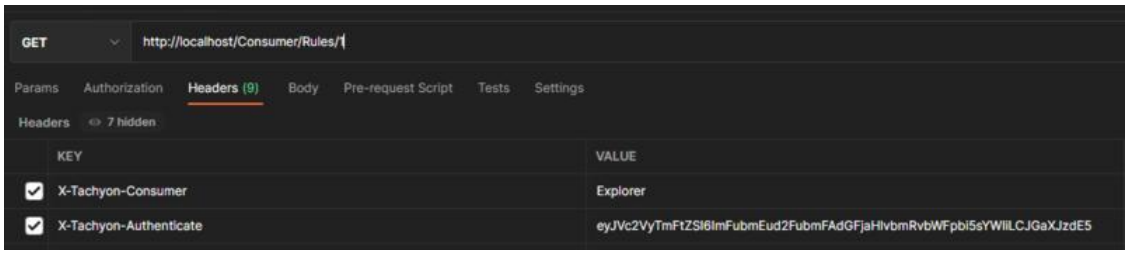
Logging out is optional and you can simply opt to "forget" the token. Information about your session will be automatically cleaned up after your token expires.

2.3.2.2.1.1.2.5 Token based authentication and OAuth support - Contacting the API

Once you have a Token you can call APIs.

To do so, set your authentication to "no authentication" and add a header field called "X-Tachyon-Authenticate" into your request. The value of this field should be the Token. Do not add any extra characters, quotes or anything else like that. The value of this field is expected to be just the token.

Here's an example of how this would look in Postman:



2.3.3 Error Conditions

If an error occurs, for example you use /Consumer/InstructionDefinitions/33333 when there are only 23 instruction definitions, the Consumer API returns a 404 error and XML error message, for example:

```
HTTP/1.1 404 Not Found
 Cache-Control: no-cache
 Pragma: no-cache
 Content-Type: application/xml; charset=utf-8
 Expires: -1
```

```
 Server: Microsoft-IIS/10.0
 X-AspNet-Version: 4.0.30319
 Persistent-Auth: true
 X-Powered-By: ASP.NET
 Date: Thu, 04 Feb 2016 11:34:59 GMT
 Content-Length: 142
{
  "Message": "No HTTP resource was found that matches the request URI ' https:/
/tachyondogfood.1e.local/Consumer/InstructionDefinitions/33333 '."
 }
```

2.3.4 Consumer API SDK assembly

From v1.7 Tachyon provides an assembly to consume Consumer API. This assembly is versioned and wraps all the consumer API end points available in a version.

2.3.5 Release Notes

2.3.5.1 Api changes between Tachyon 4.0 and 4.1

There have been no API endpoints introduced, deprecated or removed in version 4.1.

2.3.5.2 Api changes between Tachyon 4.1 and 5.0

- Consumer/PrincipalSearch method using GET verb is now **deprecated** and will be removed in a future version. Please use Consumer/PrincipalSearch using POST verb instead.
- Consumer/PrincipalSearch method using POST verb has been added to augment Active Directory user and group search with the capability to request limiting the number of results returned, define sorting and explicitly specify type of AD objects to search (users, groups or both)
- Consumer/SystemInformation/IsConsumerLicensed/Name/{name} method using GET verb has been added to allow checking consumer's licensed status by consumer name.
- Consumer/SystemInformation/IsConsumerLicensed/Id/{id} method using GET verb has been added to allow checking consumer's licensed status by consumer Id.
- Consumer/SystemInformation/LicensedConsumers method using GET verb has been added to allow retrieval of all licensed consumers.
- Consumer/EventSubscriptionAssignments API has been introduced. It has 5 methods. This API and its methods have been introduced to support eventing mechanism.
- Consumer/EventSubscriptions API has been introduced. It has 10 methods. This API and its methods have been introduced to support eventing mechanism.

2.3.6 ApplicableOperations

Entity - Role Based Access Control - Interrogates the SecurableType table in the database

Verb	Request	Notes	Permissions required
GET - by securable type id	/Consumer/ApplicableOperations/SecurableTypeId/{securableTypeId}	Gets list of all available applicable operations for the securable type by its id	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET - by securable type name	/Consumer /ApplicableOperations/SecurableTypeName/{securableTypeName}	Gets list of all available applicable operations for the securable type by its name	• Read Returns Unauthorized (401) if user does not have the required
POST	/Consumer /ApplicableOperations	Creates an applicable operation. Payload should include securable type id or name. Using both causes ambiguity in the Consumer API which will not try to set any priority over which one to use and return a	• Write Returns Unauthorized (401) if user does not

Verb	Request	Notes	Permissions required

		Bad Request (400) HTTP status.
		
		Example Request - securable type id
		
		`{ "OperationName":"Operation", "SecurableTypeId":8 }`
		
		Example Request - securable type name
		
		`{ "OperationName":"Operation", "SecurableTypeName":"SecurableType" }`
			have the required

DELETE	/Consumer /ApplicableOperations/{id}	Deletes applicable operation by id	• Write Returns Unauthorized (401) if user does not have the required

2.3.7 Approvals

Entity - Approvals - Handles instruction approval

Note for API version 8.0

As of API version 8.0, the visibility of the Instructions is restricted in accordance to the Management Groups assigned to the users. See the note at the top of the Instructions page for an explanation of how this works. The same principle applies to the Approvals API, where the approver will not be able to access or approve an instruction without having visibility on at least the same Management Groups as the user who sent the original instruction that needs to be approved.

Verb	Request	Notes	Permissions required
POST	/Consumer/Approvals	This endpoint can be used to post approval message to the API which will either approve or reject given instruction Post body: • InstructionId (int) - id of the instruction • Comment (string) - any comment to go with the approval or rejection • Approved (bool flag) - true if the instruction is to be approved or false if it is to be rejected. **Warning** This is obsolete since v3.1 to support scheduled instructions. Use /Consumer/Approvals/Instruction instead.	Approver permission on the instruction user is attempting to approve
GET	/Consumer/Approvals/CanApprove/{instructionId}	Can be used to check if calling user can approve given instruction Returns object with following fields: • CanApprove (bool flag) - true if yes, false if no • ErrorMessage (string) - actual error message	None. Internally the method calls the same code as the authorization code

Verb	Request	Notes	Permissions required
		• ErrorType - (string) for false flag denotes the type of issue encountered. Values supported so far are: – "workflow" - meaning instructions isn't in AvaitingApproval state – "ownership" - meaning the calling user is asking about his/her own action (and the override approver permission is off) – "permissions" - meaning the user failed the IsAuthenticated call either because they do not have the permission to approve or do not exist in the system at all. **Warning** This is obsolete since v3.1 to support scheduled instructions. Use /Consumer/Approvals/CanApprove/instruction/{id}instead.	but instead of returning Unauthorized code to the browser returns CanApprove = false and ErrorType = "permissions".
GET	/Consumer/Approvals/GetNotifications	Can be used to get the list of Instructions that the calling user can approve **Warning** This is obsolete since v3.1 to support scheduled instructions. Use /Consumer/Approvals/notifications/instructions instead.	Instructions are filtered based on user's permissions.

Verb	Request	Notes	Permissions required
POST	/Consumer/Approvals/Instruction	To approve or reject an instruction approval request. Post body: • InstructionId (int) - id of the instruction • Comment (string) - any comment to go with the approval or rejection • Approved (bool flag) - true if the instruction is to be approved or false if it is to be rejected. **Note** Since v3.1	Approver permission on the instruction user is attempting to approve
POST	/Consumer/Approvals/ScheduledInstruction	To approve or reject an scheduled instruction approval request. Post body: • ScheduledInstructionId(int) - id of the instruction • Comment (string) - any comment to go with the approval or rejection • Approved (bool flag) - true if the instruction is to be approved or false if it is to be rejected. **Note** Since v3.1	Approver permission on the scheduled instruction user is attempting to approve
GET	/Consumer/Approvals/notifications/instructions	Can be used to get the list of Instructions that the calling user can approve **Note**	Instructions are filters based on

Verb	Request	Notes	Permissions required
		Since v3.1	user's permissions.
GET	/Consumer/Approvals/notifications/scheduledinstructions	Can be used to get the list of scheduled Instructions that the calling user can approve **Note** Since v3.1	Scheduled instructions are filters based on user's permissions.
GET	/Consumer/Approvals/CanApprove/instruction/{id}	Can be used to check if calling user can approve given instruction Returns object with following fields: • CanApprove (bool flag) - true if yes, false if no • ErrorMessage (string) - actual error message • ErrorType - (string) for false flag denotes the type of issue encountered. Values supported so far are: – "workflow" - meaning instructions isn't in AvaitingApproval state – "ownership" - meaning the calling user is asking about his/her own action (and the override approver permission is off)	None. Internally the method calls the same code as the authorization code but instead of returning Unauthorized code to the browser returns CanApprove = false

Verb	Request	Notes	Permissions required
		– "permissions" - meaning the user failed the IsAuthenticated call either because they do not have the permission to approve or do not exist in the system at all. **Note** Since v3.1	and ErrorType = "permissions".
GET	/Consumer/Approvals/canapprove/scheduledinstruction/{id}	Can be used to check if calling user can approve given instruction Returns object with following fields: • CanApprove (bool flag) - true if yes, false if no • ErrorMessage (string) - actual error message • ErrorType - (string) for false flag denotes the type of issue encountered. Values supported so far are: – "workflow" - meaning instructions isn't in AvaitingApproval state – "ownership" - meaning the calling user is asking about his/her own action (and the override approver permission is off) – "permissions" - meaning the user failed the IsAuthenticated call either because they do not have the permission to approve	None. Internally the method calls the same code as the authorization code but instead of returning Unauthorized code to the browser returns CanApprove = false and ErrorType =

Verb	Request	Notes	Permissions required
		or do not exist in the system at all. **Note** Since v3.1	"permissions".
GET	/Consumer/Approvals/notifications	Can be used to get all approval requests that the calling user can approve. These include • Instructions • Scheduled instructions • Device authorization for two factor authentication **Sample Response Payload** `{ "Instructions": [{ "Id": 2, "Sequence": 0, "Name": "1E-Explorer-1ECore-MyInstruction1", // properties removed for brevity }, { "Id": 3, "Sequence": 0, "Name": "1E-Explorer-1ECore-MyInstruction2", // properties removed for brevity }], "ScheduledInstructions": [{ "Id": 1, "Name": "1E-Explorer-Examples-MyInstruction1", "Cmd": "SendAll", "KeepRaw": true,`	All the items are filtered based on user permissions. For example, if a user is not allowed to approve instructions then the call will return empty instructions array.

Verb	Request	Notes	Permissions required


```
            "Scope": null,
            "InstructionTtlMinutes": 60,
            "ResponseTtlMinutes": 120,
            "CreatedTimestampUtc": "2017-11-21T16:37:42.997Z",
            "Export": false,
            "ExportLocation": null,
            // properties removed for brevity
        },
        {
            "Id": 1,
            "Name": "1E-Explorer-Examples-MyInstruction2",
            "Cmd": "SendAll",
            "KeepRaw": true,
            "Scope": null,
            "InstructionTtlMinutes": 60,
            "ResponseTtlMinutes": 120,
            "CreatedTimestampUtc": "2017-11-21T16:37:42.997Z",
            "Export": false,
            "ExportLocation": null,
            // properties removed for brevity
        }
    ],
    "Devices": [
        {
            "Id": 1,
            "UserName": "acme\\jane.doe",
            "UserEmail": "jane.doe@acme.com",
            "RequestGuid": "085fb65f-f653-4336-a767-17fc826cd453",
            "DeviceType": null,
            "CreatedTimestampUtc": "2017-11-21T16:39:15.723Z",
```

Verb	Request	Notes	Permissions required

		<pre>"ModifiedTimestampUtc": "2017-11-21T16:39:15.723Z",
		"Manufacturer": null,
		"Model": null,
		"OsVersion": null
		}
		]
		}</pre> **Note** Since v3.2	

2.3.8 AuditLogs

Entity - Approvals - Handles audit log entries

Verb	Request	Notes	Permissions required
POST	/Consumer/AuditLogs	Returns audit logs that fulfil parameters passed in the request body: • filter - filter expression. To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Component • Comment	Requires 'Read' permission to the 'security' SecurableType

Verb	Request	Notes	Permissions required
		• CreatedTime • Message • UserName Allowed sort columns: • Component • Comment • CreatedTime • Message • UserName	

2.3.9 Authentication

Entity - Authentication - Handles user authentication

Verb	Request	Notes	Permissions required

POST	/Consumer/Authentication	Authenticates given user
		
		Post body:
		
		Payload example
		
		{
		"UserName": "user name",
		"Password": "pwd"
		}
			none

POST	/ConsumerAuthentication/Token	To send instruction authentication token. Used by instruction Two Factor Authentication Post body: **Payload**	Only the user creating the instruction is

Verb	Request	Notes	Permissions required

		{
		"InstructionId" : 1,
		"Token": "ABC"
		}
		Warning
		This is obsolete since v3.1 to support scheduled instructions. Use /Consumer/Authentication/Instruction/Token instead.
			allowed to call.

POST	/Consumer/Authentication/Instruction/Token	To send instruction authentication token.
		Used while doing two-factor authentication.
		Request Payload
		{
		"Id" : 1,
		"Token": "ABC"
		}
		Note
		Since v3.1
			Only the user creating the instruction is allowed to call.

POST	/Consumer/Authentication/ScheduledInstruction/Token	To send scheduled instruction authentication token.
		Used while doing two-factor authentication.
		Request Payload
		{
		"Id" : 1,
		"Token": "ABC"
		}
		Note
		Since v3.1
			Only the user creating the scheduled instruction is allowed to call.

Verb	Request	Notes	Permissions required

POST	/Consumer/Authentication/AuthorizeDevice	**API versions up to and including 3.3. Unavailable in versions 4.0 onwards.**
		
		Authorizes or rejects a mobile device to be used for two factor authentication
		
		Payload
		
		{
		"RequestId": 1,
		"RequestGuid": "",
		"Authorized" : true,
		"Comment" : ""
		}
		
		Request Id and GUID must be supplied and must belong to a valid request.
		
		Authorized should be true if the device should be authorized for 2FA or false if you wish to reject the request.
		
		Comment is optional
			Requires 'Write' permission on 'Security'

GET	/Consumer/Authentication/AuthorizationRequests?getall=true|false	**API versions up to and including 3.3. Unavailable in versions 4.0 onwards.** Returns authorization requests in the system If getall is set to true, all requests will be returned. If it is set to false only pending requests are returned. **Return payload** { "Id": 1, "UserName": "", "UserEmail": "", "RequestGuid": "", "DeviceType": "", "CreatedTimestampUtc": "	Requires 'Read' permission on 'Security'

Verb	Request	Notes	Permissions required
		", "ModifiedTimestampUtc": " ", "Manufacturer": "", "Model": "", "OsVersion": "", }	
GET	/Consumer/Authentication/AuthorizationRequests/{id}	**API versions up to and including 3.3. Unavailable in versions 4.0 onwards.** Returns a specific authorization request. Return payload as above	Requires 'Read' permission on 'Security'

2.3.10 ComponentHealthManager

Minimum API version 5.1

Web	Request	Notes	Permission Required
GET	/Consumer/ComponeneHealthManager	Returns all information about component health for the entire system	'Read' on 'Infrastructure'

2.3.11 Consumers

Entity - Consumers - Handles Consumers within the system

Verb	Request	Notes	Permissions requires
GET	/Consumer/Consumers	This endpoint can be used to retrieve all consumers in the system.	Requires

<table>
<tr><th>Verb</th><th>Request</th><th>Notes</th><th>Permissions requires</th></tr>
<tr><td></td><td></td><td>This will return a collection of objects, each of which will have following fields:
<table><tr><th>Field name</th><th>Description</th></tr><tr><td>Id</td><td>the ID of the consumer</td></tr><tr><td>Name</td><td>Name of the Consumer</td></tr><tr><td>OffloadTargetUrl</td><td>offload address: the URL where the Responses shall be POSTed (including the http:// or https:// prefix)</td></tr><tr><td>OffloadPostTimeoutSeconds</td><td>Maximum time in seconds that Tachyon will wait for the OffloadTarget to respond</td></tr><tr><td>OffloadUseWindowsAuth</td><td>boolean flag defining whether Windows Authentification will be used by the Tachyon Core when connecting to</td></tr></table></td><td>'Read' permission on 'Consumer'</td></tr>
</table>

Verb	Request	Notes		Permissions requires
			the OffloadTarget	
		ApplicationUrl	a web Url where that consumer can be found. This field is optional and is mostly applicable to web based consumers	
		IsEnabled	boolean flag defining if the consumer is enabled or not	
		SystemConsumer	boolean flag defining if this is a system consumer.	
		MaximumSimultaneousInFlightInstructions	maximum number of instructions issued by this consumer that can be 'in-flight' at any one time	
GET	/Consumer/Consumers/Id/{Id}	This endpoint can be used to retrieve a single consumer by its Id. Return object as above		Requires 'Read' permission on

Verb	Request	Notes	Permissions requires
			'Consumer'
GET	/Consumer/Consumers/Name/{Name}	This endpoint can be used to retrieve a single consumer by its Name (string must be Base64 encoded). Return object as above	Requires 'Read' permission on 'Consumer'
POST	/Consumer/Consumers/Search	**Minimum API version 4.0** Returns consumers that fulfil parameters passed in the request body: • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Name • Enabled • MaximumSimultaneousInstructions • OffloadTargetUrl • SystemConsumer Allowed sort columns:	Requires 'Read' permission on 'Consumer'

<table>
<tr><th>Verb</th><th>Request</th><th>Notes</th><th>Permissions requires</th></tr>
<tr><td></td><td></td><td>Name
Enabled
MaximumSimultaneousInstructions
SystemConsumer</td><td></td></tr>
<tr><td>POST</td><td>/Consumer/Consumers</td><td>This endpoint can be used to add a new Consumer to the system.
Post body:<table><tr><th>Field Name</th><th>Description</th><th>Required?</th><th>Notes</th></tr><tr><td>Name</td><td>Name of the Consumer</td><td>Yes</td><td>Must be unique.
Must be alpha</td></tr></table></td><td>Requires 'Write' permission on 'Consumer'</td></tr>
</table>

Verb	Request	Notes				Permissions requires
					numeric without spaces.	
		OffloadTargetUrl	Offload address: the URL where the Responses shall be POSTed (including the http:// or https:// prefix)	No	If provided must be a valid Url syntax-wise.	
		OffloadPostTimeoutSeconds	Maximum time in seconds	No		

Verb	Request	Notes				Permissions requires
			that Tachyon will wait for the OffloadTarget to respond			
		OffloadUseWindowsAuth	Boolean flag defining whether Windows Authentification will be used by the Tachyon Core when connecting to the OffloadTarget	No		
		ApplicationUrl	A web Url where that consumer can be found. This field is optional and is mostly applicabl	No	If provided must be a valid Url syntax-wise.	

Verb	Request	Notes				Permissions requires
			e to web based consumers			
		IsEnabled	Boolean flag defining if the consumer is enabled or not	No	If not provided will default to false, which will mean the consumer will be disabled.	
		MaximumSimultaneousInFlightInstructions	Maximum number of instructions issued by this consumer that can be 'in-flight' at any one time	Yes	Must be lower than global limit defined in the GlobalSettings table in the database.	
PUT	/Consumer/Consumers	This endpoint can be used to modify an existing Consumer within the system. Put body:				Requires 'Write' permission

<table>
<tr><th>Verb</th><th>Request</th><th>Notes</th><th></th><th></th><th></th><th>Permissions required on 'Consumer'</th></tr>
<tr><td></td><td></td><td></td><td>Description</td><td>Required?</td><td>Notes</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Id</td><td>Id of the Consumer to be modified</td><td>Yes</td><td>Must be a valid consumer Id</td></tr>
<tr><td></td><td></td><td></td><td>Name</td><td>Name of the Consumer</td><td>Yes</td><td>• Must be unique.
• Must be alpha numeric without spaces.</td></tr>
<tr><td></td><td></td><td></td><td>OffloadTargetUrl</td><td>Offload address: the URL where the Responses shall be POSTed (including the http:// or</td><td>No</td><td>If provided must be a valid Url syntax-wise.</td></tr>
</table>

Verb	Request	Notes				Permissions requires
			https:// prefix)			
		OffloadPostTimeoutSeconds	Maximum time in seconds that Tachyon will wait for the OffloadTarget to respond	No		
		OffloadUseWindowsAuth	Boolean flag defining whether Windows Authentification will be used by the Tachyon Core when connecting to the OffloadTarget	No		
		ApplicationUrl	A web Url where that	No	If provided must be a valid Url	

Verb	Request	Notes				Permissions requires
			consumer can be found. This field is optional and is mostly applicable to web based consumers		syntax-wise.	
		IsEnabled	Boolean flag defining if the consumer is enabled or not	No	If not provided will default to false, which will mean the consumer will be disabled.	
		MaximumSimultaneousInFlightInstructions	Maximum number of instructions issued by this consumer that can be 'in-flight' at	Yes*	• This value must be provided for non-system	

Verb	Request	Notes			Permissions requires
			any one time	consumers. • Must be lower than global limit defined in the GlobalSettings table in the database.	
DELETE	/Consumer/Consumers/{Id}	Deletes the consumer. Preconditions: The Consumer cannot be in use by an Instruction or a Scheduled Instruction. System Consumers cannot be deleted.			Requires 'Write' permission on 'Consumer'

Verb	Request	Notes	Permissions requires
DELETE	/Consumer/Consumers	**Minimum API version 4.0** Deletes multiple consumers. System consumers cannot be deleted. Consumers with scheduled instructions cannot be deleted. Body of the request should contain an array of Consumer Ids.	Requires 'Write' permission on 'Consumer'

2.3.12 CustomProperties

Entity - Custom property

Verb	Request	Notes	Permissions required
GET - for property type by id	/Consumer/CustomProperties/TypeId/{typeId}	Gets list of all available custom properties for a type by its id	Following product pack related permissions • Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if

Verb	Request	Notes	Permissions required
			user does not have the required permissions.
GET - for property type by name	/Consumer/CustomProperties/TypeName/{typeName}	Gets list of all available custom properties for a type by its name	Following product pack related permissions • Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have the required permissions.
GET - individual property by its id	/Consumer/CustomProperties/Id/{Id}	Gets an individual custom property by id	Following product pack related permissions • Questioner

Verb	Request	Notes	Permissions required
			• Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have the required permissions.
POST	/Consumer/CustomProperties/Search	**Minimum API version 4.0** Returns custom properties that fulfil parameters passed in the request body: • filter - filter expression.To learn about how to define filter checkUsing scope and filter expressionspage. • sort - sort expression. To learn about how to define sort criteria checkSort Definitionpage. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch	Following custom property related permission • Read

Verb	Request	Notes	Permissions required
		Allowed filter columns: • Name Allowed sort columns: • Name	

POST	/Consumer/CustomProperties	Creates a custom property.
		Payload should include custom property type id or name. Using both causes ambiguity in the Consumer API which will not try to set any priority over which one to use and return a Bad Request (400) HTTP status.
		Example Request - custom property type id
		`{ "Name":"Business Unit", "TypeId":1 }`
		Example Request - custom property type name
		`{ "Name":"Business Unit", "TypeName":"ScopableTag" }`
		From API version 4.0 onwards, you can add a collection of values that should be added for the property being created:
		Example Request - custom property type name
			• Write
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required

		<pre>{
		"Name":"Business Unit
		",
		"TypeName":"ScopableTa
		g",
		"Values":[{
		"Value":"First"
		},
		{
		"Value":"Second"
		}]
		}</pre>	

PUT	/Consumer/CustomProperties	Updates a custom property. Only name can be updated.
		
		Update Payload
		
		<pre>{
		"Id":7,
		"Name": "Building"
		}</pre>
		From API version 4.0 you can provide values that should replace any values the custom property had before the update. This is done via "Values" property in the payload as seen below:
		
		Example Request - custom property type name
		
		<pre>{
		"Id":"1",
		"Name":"Business Unit
		",
		"TypeName":"ScopableTa
		g",
		"Values":[{
		"Value":"First"
		},
		{
		"Value":"Second"</pre>
			• Write
			
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		}] } • If "Values" property is omitted or set to null, custom property values will not be modified • If "Values" property is set to an empty array (so [] is JSON), all custom property values belonging to this custom property will be removed. • If "Values" property contains a collection of values, they will replace any values that the custom property being updated had before the update.	
DELETE	/Consumer/CustomProperties/{id}	Deletes a custom property by id	• Write Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/CustomProperties	**Minimum API version 4.0** Deletes multiple custom properties. Body of the request should contain an array of custom	• Write Returns Unauthorized (401) if user does not have the

Verb	Request	Notes	Permissions required
		property Ids for properties that should be deleted.	required permissions.

2.3.13 CustomPropertyTypes

Entity - Custom property type

Verb	Request	Notes	Permissions required
GET - All	/Consumer/CustomPropertyTypes	Gets list of all available custom property types	Following product pack related permissions • Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have the required permissions.
GET - individual by id	/Consumer/CustomPropertyTypes/Id/{id}	Gets individual by id	Following product pack related permissions

Verb	Request	Notes	Permissions required
			• Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have the required permissions.
GET - individual by name	/Consumer/CustomPropertyTypes/Name/{name}	Gets individual by name	Following product pack related permissions • Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have

<table>
<tr><th>Verb</th><th>Request</th><th>Notes</th><th>Permissions required</th></tr>
<tr><td></td><td></td><td></td><td>the required permissions.</td></tr>
<tr><td>POST</td><td>/Consumer/CustomPropertyTypes</td><td>Minimum API version 4.0

Creates a new custom property type

Only non-system custom property types can be created.

<table><tr><th>Field</th><th>Description</th></tr><tr><td>Name</td><td>Name of the custom property type</td></tr><tr><td>Description</td><td>Description of the custom property type</td></tr></table></td><td>Requires 'Write' permissions on 'CustomProperty' securable type</td></tr>
<tr><td>PUT</td><td>/Consumer/CustomPropertyTypes</td><td>Minimum API version 4.0

Updates an existing custom property type.

System custom property types cannot be modified.</td><td>Requires 'Write' permissions on 'CustomProperty' securable type</td></tr>
</table>

Verb	Request	Notes		Permissions required
		Field	Description	
		Id	Id of the custom property type to be updated	
		Name	Name of the custom property type	
		Description	Description of the custom property type	
DELETE	/Consumer/CustomPropertyTypes/Id/{id}	**Minimum API version 4.0** Deletes an individual custom property type by its id. System custom property types cannot be deleted. Deleting a custom property type will cause all custom properties of this type and all of their values to be deleted.		Requires 'Write' permissions on 'CustomProperty' securable type

Verb	Request	Notes	Permissions required
DELETE	/Consumer/CustomPropertyTypes/Name/{name}	**Minimum API version 4.0** Deletes an individual custom property type by its name. System custom property types cannot be deleted. Deleting a custom property type will cause all custom properties of this type and all of their values to be deleted.	Requires 'Write' permissions on 'CustomProperty' securable type
DELETE	/Consumer/CustomPropertyTypes	**Minimum API version 4.0** Deletes multiple custom property types. System custom property types cannot be deleted. Body of the request should contain an array of custom property types	Requires 'Write' permissions on 'CustomProperty' securable type

2.3.14 CustomPropertyValues

Entity - Custom property value

Verb	Request	Notes	Permissions required
GET - for	/Consumer/CustomPropertyValues/Property/{propertyId}	Gets list of all available custom	Following product pack

Verb	Request	Notes	Permissions required
property by id		property values for a property by its id	related permissions • Questioner • Actioner Following custom property related permission • Read Returns Unauthorized (401) if user does not have the required permissions.
GET - individual property value by its id	/Consumer/CustomPropertyValues /Id/{Id}	Gets an individual custom property value by id	Following product pack related permissions • Questioner • Actioner Following custom property related permission

Verb	Request	Notes	Permissions required
			• Read Returns Unauthorized (401) if user does not have the required permissions.
POST	/Consumer/CustomPropertyValues	Creates an custom property value. **Example Request** `{ "Value": "Marketing", "PropertyId":1 }`	Following custom property related permission • Write Returns Unauthorized (401) if user does not have the required permissions.
POST	/Consumer/CustomPropertyValues/Property/{propertyId}/ReplaceValues	**Minimum API version 4.0** Replaces all values for given custom property with the values passed in this call. propertyId (URI parameter) - id of the custom property whose values are to be replaced	Following custom property related permission • Write Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		values (BODY parameter) - an array of strings that should become the new values for the custom property	
DELETE	/Consumer/CustomPropertyValues/{id}	Deletes an custom property value by id	Following custom property related permission • Write Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/CustomPropertyValues	**Minimum API version 4.0** Deletes multiple custom property values. Body of the request should contain an array of custom property value ids.	Following custom property related permission • Write Returns Unauthorized (401) if user does not have the required permissions.

2.3.15 DataExport

Entity - Data Export - Exports information from the system

Note for API version 8.0

As of API version 8.0, the visibility of the instruction responses is restricted in accordance to the Management Groups assigned to the users. See the note at the top of the Responses page for an explanation of how this works. The same principle applies to the DataExport APIs that export responses, where it is not possible to export any responses that cannot be accessed from the Responses API.

Verb	Request	Notes	Permissions required
POST	/Consumer/DataExport	This endpoint can be used to convert any data formatted as a JSON array into a CSV. This endpoint provides a service and does NOT expose any internal data of the Tachyon system • Post using x-www-form-urlencoded form with following fields: • FileName: Name of the CSV file you wish to receive back • Data: JSON array containing the data you wish to convert into CSV	Usable by users that have any permissions on any of the product packs. • Viewer • Questioner • Actioner • Approver Returns Unauthorized (401) if user does not have the required permissions.
POST	/Consumer/DataExport/ExportResponses	Exports responses for a specified question. Send JSON as body of the message with following fields: • InstructionId - id of the instruction for	Any permission on the instruction whose responses are to be exported.

Verb	Request	Notes	Permissions required
		which you wish to export responses • ExportPath - path where the results file is to be placed	
POST	/Consumer/DataExport/ExportOtherResponses	**Minimum API version 3.2** Exports other responses for a specified question. Send JSON as body of the message with following fields: • InstructionId - id of the instruction for which you wish to export responses • ExportPath - path where the results file is to be placed	Any permission on the instruction whose responses are to be exported.
GET	/Consumer/DataExport/Type/{objectType}	**UNAVAILABLE FROM 2.1 (INC) ONWARDS** This endpoint can be used to retrieve objects of specific type from the Tachyon system. Supported object type values at the time of writing are: "devices", "instructiondefinitions", "questiondefinitions", "actiondefinitions" and "instructions"	Only **"Global Administrators"** role. Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
GET	/Consumer/DataExport/Responses/{instructionId:int}/aggregate/{flag:bool}	**UNAVAILABLE FROM 2.1 (INC) ONWARDS** This endpoint can be used to retrieve successful responses to a specific instruction. • instructionId - id of the instruction • flag - "true" if you want aggregated responses. "false" if you don't. NOTE - True will only work for questions that you can aggregate on the way out (so the ones with "Keep results") and only those will return a value here. Others will simply return an empty file.	Usable by users who have view permission to given instruction (same as Instructions endpoints) • Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.
GET	/Consumer/DataExport/ResponsErrors/{instructionId:int}/aggregate/{flag:bool}	**UNAVAILABLE FROM 2.1 (INC) ONWARDS** This endpoint can be used to errors (or other results) to a specific instruction.	Usable by users who have view permission to given instruction (same as Instructions endpoints)

Verb	Request	Notes	Permissions required
		• instructionId - id of the instruction • flag - "true" if you want aggregated responses. "false" if you don't.	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.

2.3.16 Deploy

Entity - Deployment - Interrogates the AgentDeploymentJob and AgentDeploymentDevice tables

Verb	Request	Notes	Permissions required
GET	Consumer/Deploy/Jobs	Gets all stored Deployment Jobs. **Returned JSON** `[ { "Id": 1, "Description": "First job", "Comments": "", "Status": 7,`	View on AgentDeployment

Verb	Request	Notes	Permissions required


```
        "WorkflowState": 8,
        "CreatedBy": "somed
omain\\someuser",
        "CreatedTimestampUt
c": "2017-11-20T15:49:07.31
Z",
        "Credentials": {
            "UserName": "so
meuser",
            "Password": "AQ
AAANC..."
        },
        "CallbackUrl": nul
l,
        "ConsumerCustomData
": null,
        "AgentSettings": {
            "SWITCH": "loca
lhost",
            "BACKGROUNDCHAN
NELURL": "http://localhost/
background/"
        }
    },
    {
        "Id": 2,
        "Description": "Sec
ond job",
        "Comments": "",
        "Status": 10,
        "WorkflowState": 9,
        "CreatedBy": "somed
omain\\someuser",
        "CreatedTimestampUt
c": "2017-11-20T15:49:57.1Z
",
        "Credentials": {
            "UserName": "so
meuser",
            "Password": "AQ
AAANC..."
        },
        "CallbackUrl": nul
l,
        "ConsumerCustomData
": null,
        "AgentSettings": {
```

Verb	Request	Notes	Permissions required

		<pre>"SWITCH": "localhost",
		"BACKGROUNDCHANNELURL": "http://localhost/background/"
		}
		},
		{
		"Id": 3,
		"Description": "another one",
		"Comments": "",
		"Status": 10,
		"WorkflowState": 9,
		"CreatedBy": "somedomain\\someuser",
		"CreatedTimestampUtc": "2017-11-20T15:55:25.663Z",
		"Credentials": {
		"UserName": "someuser",
		"Password": "AQAAANC..."
		},
		"CallbackUrl": null,
		"ConsumerCustomData": null,
		"AgentSettings": {
		"SWITCH": "Tachyon.Acme.local",
		"BACKGROUNDCHANNELURL": "http://Tachyon.Acme.local/background/"
		}
		}
		]</pre>
		Please note all passwords received inside the Credentials object will be encrypted	

Verb	Request	Notes	Permissions required
POST	/Consumer/Deploy/Jobs/Search	Gets Deployment Jobs, with options to filter, sort and paginate. **Example Request** `{ "filter":{ "Operator": "AND", "Operands": [ { "Attribute": "Status", "Operator": "!=", "Value": "0" } ] }, "start": 1, "pageSize": 10, "sort": [ { "Column": "CreatedTime", "Direction": "DESC" } ] }` • filter - filter expression. To learn about how to define filter check Using scope and filter expressions page. • start - starting index to support pagination. Begins from 1. • pagesize - number of results to fetch • sort - sort expression. To learn about how to define	View on AgentDeployment

Verb	Request	Notes	Permissions required

sort criteria check Sort Definition page.

Allowed filter columns:

- Id
- Description
- Comments
- Status
- WorkflowState
- ConsumerCustomData
- CreatedBy
- CreatedTime

Allowed sortable columns

- Id
- Description
- Comments
- Status
- WorkflowState
- ConsumerCustomData
- CreatedBy
- CreatedTime

Example Response

```
[
    {
        "Id": 1,
        "Description": "Tes
t job One",
        "Comments": "This i
s comment One",
```

Verb	Request	Notes	Permissions required

		<pre>"WorkflowState": 0,
		"CreatedBy": "ACME\
		\Admin",
		"CreatedTimestampUt
		c": "2017-10-31T14:45:01.39
		7Z",
		"Credentials": {
		"UserName": "Te
		stUser",
		"Password": "AQ
		AAANCMnd8BFdER...RYRtIG9dFc
		Y="
		},
		"CallbackUrl": nul
		l,
		"ConsumerCustomData
		": null,
		"CallbackUrl": nul
		l,
		"ConsumerCustomData
		": null,
		"AgentSettings": {
		"SWITCH": "TACH
		YONAPP.ACME.LOCAL:4000",
		"BACKGROUNDCHAN
		NELURL": "https://Tachyon.A
		CME.LOCAL:443/Background/"
		}
		}
		]</pre>	

GET	/Consumer/Deploy/Jobs/{id}	Retrieves one Job, identified by its id
		
		Example request: GET ... /Consumer/Deploy/Jobs/1
		
		Example Response
		
		<pre>{
		"Id": 1,
		"Description": "Test jo
		b One",
		"Comments": "This is co
		mment One",
		"WorkflowState": 0,
		"CreatedBy": "ACME\\Adm</pre>
			View on AgentDeployment

Verb	Request	Notes	Permissions required

		in",
		"CreatedTimestampUtc":
		"2017-10-31T14:45:01.397Z",
		"Credentials": {
		"UserName": "TestUs
		er",
		"Password": "AQAAAN
		CMnd8BFdER...RYRtIG9dFcY="
		},
		"CallbackUrl": null,
		"ConsumerCustomData": n
		ull,
		"CallbackUrl": null,
		"ConsumerCustomData": n
		ull,
		"AgentSettings": {
		"SWITCH": "TACHYONA
		PP.ACME.LOCAL:4000",
		"BACKGROUNDCHANNELU
		RL": "https://Tachyon.ACME.
		LOCAL:443/Background/"
		}
		}
		
		Please note all passwords received inside the Credentials object will be encrypted	

POST	/Consumer/Deploy/Devices/Search	Retrieves all deployment devices that satisfy the conditions specified by a filter. Options to sort and paginate.
		
		Example Request
		
		{
		"filter":{
		"Operator": "AND",
		"Operands": [
		{
		"Attribute
		": "InstallStatus",
		"Operator":
		"!=",
		"Value": "1
		9"
			View on AgentDeployment

Verb	Request	Notes	Permissions required

```
                }
            ]
        },
        "start": 1,
        "pageSize": 10,
        "sort": [
            {
                "Column": "Id",
                "Direction": "A
SC"
            }
        ]
}
```

- filter - filter expression. To learn about how to define filter check Using scope and filter expressions page.
- start - starting index to support pagination. Begins from 1.
- pagesize - number of results to fetch
- sort - sort expression. To learn about how to define sort criteria check Sort Definition page.

Allowed filter columns:

- Id
- JobId
- Address
- OperatingSystem
- OperatingSystemArchitecture
- TachyonAgentVersion

Verb	Request	Notes	Permissions required

- InstallStatus
- StatusMessage
- ConsumerCustomData

Allowed sortable columns

- Id
- JobId
- Address
- OperatingSystem
- OperatingSystemArchitecture
- TachyonAgentVersion
- InstallStatus
- StatusMessage
- ConsumerCustomData

Example Response


```
[
  {
    "Id": 3,
    "JobId": 3,
    "Address": "192.168.22.
2",
    "OperatingSystem": 1,
    "OperatingSystemVariant
": 1,
    "TachyonAgentVersion":
"3.1",
    "InstallStatus": 18,
    "StatusMessage": "Copyi
ng binaries failed with fol
lowing exception: .... \r\
n ....",
    "ConsumerCustomData": "
ref_1234"
  },
```

Verb	Request	Notes	Permissions required

		<pre>{
		"Id": 8,
		"JobId": 3,
		"Address": "192.168.22.66",
		"OperatingSystem": 1,
		"OperatingSystemVariant": 1,
		"TachyonAgentVersion": "3.1",
		"InstallStatus": 19,
		"StatusMessage": "....",
		"ConsumerCustomData": "ref_9876"
		}
		]</pre>	

GET	/Consumer/Deploy/Devices/{id}	Retrieves one deployment device, identified by its Id
		
		Example request: GET .../Consumer/Deploy/Devices/3
		
		Example Response
		
		<pre>{
		"Id": 3,
		"JobId": 3,
		"Address": "192.168.22.2",
		"OperatingSystem": 1,
		"OperatingSystemVariant": 1,
		"TachyonAgentVersion": "3.1",
		"InstallStatus": 18,
		"StatusMessage": "Copying binaries failed with following exception: .... \r\n ....",
		"ConsumerCustomData": "ref_1234"
		}</pre>
			View on AgentDeployment

Verb	Request	Notes	Permissions required

POST	/Consumer/Deploy/Jobs	Creates and launches a deployment Job, including all its Devices.
		
		Example Request
		
		{
		"Job":{
		"Description": "Deploy Windows PCs on Subnet 1",
		"Comments": "This is a test job",
		"Credentials": {
		"UserName": "Admin1",
		"Password": "1234"
		},
		"CallbackUrl": "https://myClient/myEndpoint",
		"ConsumerCustomData": "myReference",
		"AgentSettings": {
		"SWITCH": "TACHYONAPP.ACME.LOCAL:4000",
		"BACKGROUNDCHANNELURL": "https://Tachyon.ACME.LOCAL:443/Background/"
		}
		},
		"Devices": [
		{
		"Address": "192.168.0.123",
		"OperatingSystem": 1,
		"OperatingSystemArchitecture": 1,
		"TachyonAgentVersion": "3.1",
		"ConsumerCustomData": "ref_1234"
		},
		{
		"Address": "cli
			Execute on AgentDeployment

Verb	Request	Notes	Permissions required
		ent1.acme.local", "OperatingSystem": 1, "OperatingSystemArchitecture": 1, "TachyonAgentVersion": "3.2" }] } **Example Response** [Returns the same information that was posted, but the generated IDs are included and the passwords encoded.]	
POST	/Consumer/Deploy/Jobs/{id}/Cancel	Cancels an existing Job. Example request: /Consumer/Deploy/Jobs/1/Cancel Returns a status of 200-OK if the cancellation request is successfully posted. Otherwise, an HTTP error status and message.	Execute on Agentdeployment
GET	/Consumer/Deploy/Configuration	Retrieves configuration information from the server. A consumer that intends to create a Deployment Job can use this information to display options to the user so that they can select and submit adequate selections for the Job. **Example Response** { "Switches": [{ "Id": 1,	Execute on Agentdeployment

Verb	Request	Notes	Permissions required

		<pre>"Name": "TACHYONAPP.ACME.LOCAL",
		"Port": 4000
		},
		{
		"Id": 2,
		"Name": "TACHYONAPP2.ACME.LOCAL",
		"Port": 4000
		}
		],
		"BackgroundChannels": [
		{
		"Id": 1,
		"ResourceUrl": "https://Tachyon.ACME.LOCAL:443/Background/"
		}
		]
		}</pre>	

GET	/Consumer/Deploy/Installers	Gets a list of all the installers currently stored in the Tachyon database.
		
		Example Response
		
		<pre>[
		{
		"Id": 5,
		"OperatingSystemType": 1,
		"OperatingSystemVersion": null,
		"OperatingSystemArchitecture": 2,
		"TachyonAgentVersion": "3.2.0.389",
		"Location": "/Installers/Windows/X64/3.2.0.389/Tachyon-x64.msi",
		"SHA256Checksum": "QtqwT4AyCZxMjRHAZlPCMT7cBW5cTIzVofLE2XCeSV8="
		},
		{</pre>
			Read on AgentInstallerManagement

Verb	Request	Notes	Permissions required

		"Id": 6,
		"OperatingSystemType": 6,
		"OperatingSystemVersion": "7.1",
		"OperatingSystemArchitecture": 2,
		"TachyonAgentVersion": "3.2.0.303",
		"Location": "/Installers/RedHat/7.1/X64/3.2.0.303/tachyon.agent-RHEL_7.1_x64_v3.2.0.303.rpm",
		"SHA256Checksum": "ojlcplO7+Elffbom9x4DoClIox/MyXlT8pWwFRacnA4="
		},
		{
		"Id": 7,
		"OperatingSystemType": 1,
		"OperatingSystemVersion": null,
		"OperatingSystemArchitecture": 1,
		"TachyonAgentVersion": "3.2.0.389",
		"Location": "/Installers/Windows/X86/3.2.0.389/Tachyon-x86.msi",
		"SHA256Checksum": "NADe5IAngOA8kZvt/DY0S3txljsEIuVL4U+RwxaD8AY="
		}
		]
		
		The values for OperatingSystemType and OperatingSystemArchitecture are shown below under "Enumerations".	

POST	/Consumer/Deploy/Installers/Search	**Minimum API version 4.0**	Read on AgentInstallerManagement

Verb	Request	Notes	Permissions required
		Returns agent installers that fulfil parameters passed in the request body: • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Id • OperatingSystemType • OperatingSystemArchitecture • OperatingSystemVersion • TachyonAgentVersion Comments Allowed sort columns: • Id • OperatingSystemType • OperatingSystemArchitecture	

<table>
<tr><th>Verb</th><th>Request</th><th>Notes</th><th>Permissions required</th></tr>
<tr><td></td><td></td><td>OperatingSystemVersion
TachyonAgentVersion</td><td></td></tr>
<tr><td>POST</td><td>/Consumer/Deploy/Installers</td><td>Submits a file to be stored in the Installer library.

This endpoint expects multipart form data with a single file attached.

The file should be the installer you wish to upload.

The form data should have following fields:

<table><tr><th>Field name</th><th>Description</th><th>Notes</th></tr><tr><td>AgentVersion</td><td>Version of the Tachyon agent this installer is for</td><td>The is a free form text field. Maximum 20 characters.</td></tr><tr><td>Os</td><td>Operating system this installer is for</td><td>This should be a textual or numerical value from the operating system enumeration</td></tr></table></td><td>Add on AgentInstallerManagement</td></tr>
</table>

Verb	Request	Notes			Permissions required
				found below.	
		OsArch	Operating system architecture this installer is for	This should be a textual or numerical value from the operating system architecture enumerating found below.	
		OsVersion	Version of the operating system this installer is for	Optional. Not used for Windows installers. This is a free form text field. Maximum of 20 characters.	
		Comments	Comments for the installer	Optional. Available from API version	

Verb	Request	Notes		Permissions required
			3.4 onwards. This is a free form text field. Maximum of 1024 characters.	
		Please note that passing 'Unknown' as operating system or operating system architecture is not allowed when uploading or deleting installers.		
PUT	/Consumer/Deploy/Installers	**Minimum API version 4.0** Updates an existing installer **Sample request body** `{ "Id": 12, "Comments":"This is an updated comment" }`		Add on AgentInstallerManagement
DELETE	/Consumer/Deploy/Installers/{id}	Deletes an installer specified by its {id}		Delete on AgentInstallerManagement
DELETE	/Consumer/Deploy/Installers	**Minimum API version 4.0** Deletes multiple installers by their ids.		Delete on AgentInstallerManagement

Verb	Request	Notes	Permissions required
		Body of this request should contain an array of Ids of the installers that should be deleted.	
DELETE	/Consumer/Deploy/Installers/AgentVersion/{agentVersion} /Os/{operatingSystemType} /OsVersion/{operatingSystemVersion} /OsArch/{operatingSystemArchitecture} -or- /Consumer/Deploy/Installers/AgentVersion/{agentVersion} /Os/{operatingSystemType} /OsArch/{operatingSystemArchitecture}	Deletes an installer specified by its characteristics. See the "POST" entry for an explanation of each parameter. Please note that passing 'Unknown' as operating system or operating system architecture is not allowed when uploading or deleting installers.	Delete on AgentInstallerManagement

GET	/Consumer/Deploy/SupportedOs	Returns a dictionary of operating systems supported by the feature.
		
		The Key is the numeric value of the operating system while the value is the textual representation of it.
		
		Sample response
		
		{
		"1": "Windows"
		}
			Read on AgentInstallerManagement

GET	/Consumer/Deploy/SupportedInstallerTypes	Returns a list of installer types that are supported to be uploaded to Tachyon. Each installer type has name and its associated extension. **Response Payload**	Read on AgentInstallerManagement

Verb	Request	Notes	Permissions required

		<pre>[
		    {
		        "Name": "Windows Installer Package",
		        "Extension": ".msi"
		    },
		    {
		        "Name": "RedHat Package Manager",
		        "Extension": ".rpm"
		    }
		]</pre>	

2.3.16.1 Enumerations

2.3.16.1.1 Operating System Type

Text value	Numerical value
Unknown	0
Windows	1
MacOs	2
CentOS	3
Fedora	4
openSuse	5
RedHat	6
SLES	7
Ubuntu	9
Debian	10

2.3.16.1.2 Operating System Architecture

Text value	Numerical value
Unknown	0
X86	1
X64	2

2.3.17 Devices

Entity - Devices - Interrogates the Devices table in the database

General note for version 8.0

As of API version 8.0, all the Device APIs "hide" from the caller any devices on which the caller does not have any assignment through Management Groups. This is intended for delegation of administration where the user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see the devices assigned to him or her, and not any other endpoints in the organization. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results are the ones that you would get if the only devices connected to the system were those on which you have any Management Group assignment.

If you need to bypass this restriction for compatibility with earlier versions, it is sufficient to assign to the user any Role on the "All Devices" Management Group. The Role does not need any specific permissions; the mere fact that the user has a Role on All Devices is sufficient for letting the user see all the devices.

Verb	Request	Notes	Permissions required
GET	/Consumer/Devices/fqdn/{fqdn}	Gets a single device based on FQDN. FQDN has to be base64 encoded.	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver)

Verb	Request	Notes	Permissions required
			on any instruction set

POST	/Consumer/Devices/fqdn	Gets summarized affected device counts given a list of devices.
		
		Example Request
		
		{
		"Scope":"device1,device2"
		}
			Requires any of the instruction-related permissions (so
			
			Viewer, Questioner, Actioner or Approver)
			
			on any instruction set

POST	/Consumer/Devices/fqdnfiltered/{instructionDefinitionId}	Gets summarized device counts given a list of devices in the same way as the Devices/fqdn API described above, but considers only those devices that are reachable by the calling user in accordance with the permissions granted by Management Groups for the specified instruction definition. **Example Request** { "Scope":"device	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any instruction set

Verb	Request	Notes	Permissions required
		1,device2" }	
GET	/Consumer/Devices	Gets all devices. Returns a list of Device objects. **Example Response** { "Id":1, "Name":"somemachine01", "Fqdn":"somemachine01.somadomain.com", "Status":1, "OsType":"Windows", "OsVerNum":2814750438195200, "OsVerTxt":"Microsoft Windows 10 Enterprise", "AgentVersion":281483566841860, "MAC":"00-15-5D-0B-AF-2B", "Manufacturer":"Dell Inc.", "ChassisType":7, "DeviceType":"Desktop", "CpuType":"Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz", "CpuArchitecture":"x64", "OsArchitecture":"x64", "RamMB":32693, "SMBiosGuid":"4c4c4544-0051-5a10-8058-b7c04f47354a", "OsGuid":"4c4c4	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any instruction set

Verb	Request	Notes	Permissions required
		544-0051-5a10-8058-b7c04f47354a", "TachyonGuid":"4c4c4544-0051-5a10-8058-b7c04f47354a", "BiosVersion":"n/a", "LastBootUTC":1454408472, "LastConnUtc":1454604670, "CreatedUtc":"2016-02-03T17:20:55.003", "VrPlatform":"", "TimeZone":0, "CertType":null, "CertExpiryUtc":null, "PushToken":null, "Model":"Precision T5600", "Domain":"Somedomain", "Tags": "||", "ConnectionState": ["SkipDigitalSignatures"], "LocalIpAddress": "192.168.0.1", "ConnectingIpAddress": "192.124.249.5", "TimeZoneId": "Europe/London", "SerialNumber": "FRBNJ333", "Criticality": 0, "Location": "1E office in Ealing, Third floor.",	

Verb	Request	Notes	Permissions required

		<pre>"CoverageTags": {},
		"MAC": "00-15-5d-0b-0c-20",
		"ConnectingIpAddress": "172.0.0.103"
		}</pre>	

POST	/Consumer/Devices/Scope	Gets devices based on static scope. It does not support sorting and pagination.
		Example Request
		<pre>{
		"Operator": "and",
		"Operands":[
		{
		"Attribute":"devtype",
		"Operator":"==",
		"Value":"Desktop"
		},
		{
		"Attribute":"ostype",
		"Operator":"==",
		"Value":"Windows"
		}
		]
		}</pre>To learn about how to define filter check Using scope and filter expressions page.
			Requires any of the instruction-related permissions (so
			Viewer, Questioner, Actioner or Approver)
			on any instruction set

Verb	Request	Notes	Permissions required
		For the list of allowed scope attributes check Defining the scope page.	
POST	/Consumer/Devices/ApproxTarget	**Obsolete in API versions 3.3 and onwards.** Gets approximate number of devices matching the scope in aggregated format. **Example Request** `{ "Operator": "and", "Operands":[ { "Attribute":"devtype", "Operator":"==", "Value":"Desktop" }, { "Attribute":"ostype", "Operator":"==", "Value":"Windows" } ] }` To learn about how to define filter check Using scope and filter expressions page. For the list of allowed scope attributes check Defining the scope page.	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any of the product packs

Verb	Request	Notes	Permissions required
POST	/Consumer/Devices/ApproxTarget/{id}	**Minimum API version 3.3.** Gets approximate number of devices matching the scope in aggregated format. Id is an Id of the instruction definition in whose context you wish to get the approximate data. This is required as permission on instructions that limit management group usage will affect what machines are targeted. **Example Request** `{ "Operator": "and", "Operands":[ { "Attribute":"devtype", "Operator":"==", "Value":"Desktop" }, { "Attribute":"ostype", "Operator":"==", "Value":"Windows" } ] }`	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any of the product packs

Verb	Request	Notes	Permissions required
		To learn about how to define filter check Using scope and filter expressions page. For the list of allowed scope attributes check Defining the scope page.	
POST	/Consumer/Devices/ApproxTarget/{instructionDefinitionId}/parent/{parentInstructionId}	**Minimum API version 3.3.** Gets approximate number of devices matching the scope in aggregated format. Should be used for follow-up instructions, because it will take into account permissions inherited from parent instruction and current permissions for the instruction definition to be used to issue an instruction. instructionDefinitionId - is an Id of the instruction definition in whose context you wish to get the approximate data. This is required as permission on instructions that limit management group usage will affect what machines are targeted. parentInstructionId - id an Id of the parent instruction to whom a	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any of the product packs

Verb	Request	Notes	Permissions required

		follow-up instruction you wish to check impact for.
		Example Request
		<pre>{
		"Operator": "and",
		"Operands":[
		{
		"Attribute":"devtype",
		"Operator":"==",
		"Value":"Desktop"
		},
		{
		"Attribute":"ostype",
		"Operator":"==",
		"Value":"Windows"
		}
		]
		}</pre>To learn about how to define filter check Using scope and filter expressions page.
		For the list of allowed scope attributes check Defining the scope page.	

GET	/Consumer/Devices/Tachyonguid/{guid}	Gets a single device by Tachyon GUID This will return 1 device as it assumes that GUID is unique.	Requires any of the instruction-related permissions (so

Verb	Request	Notes	Permissions required
		Endpoint will throw if it cannot find the device or a malformed GUID was supplied. The GUID string is to be supplied in plain text, not base64 encoded. The GUID can be in any format that is acceptable to the .NET's GUID class.	Viewer, Questioner, Actioner or Approver) on any instruction set
POST	/Consumer/Devices	This endpoint allows searching for devices based on criteria along with sorting and pagination. **Example Request** `{ "filter":{ "Operator": "and", "Operands":[ { "Attribute":"DeviceType", "Operator":"==", "Value":"Desktop" }, { "Attribute":"OSType", "Operator":"==", "Value":"Windows" } ]`	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver) on any instruction set

Verb	Request	Notes	Permissions required

```
    },
   "start":10,
   "pageSize":100,
   "sort":[
      {
         "Column
":"fqdn",
         "Direct
ion":"ASC"
      }
   ]
}
```

- filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.
- sort - sort expression. To learn about how to define sort criteria check Sort Definition page.
- start - starting index to support pagination. It begins from 1.
- pagesize - number of results to fetch

Allowed filter columns

- Name
- FQDN
- DeviceType

Verb	Request	Notes	Permissions required
		• Status • OsType • OsVersionText • AgentVersion • TimeZone • VrPlatform (Virtualization Platform) • Domain • OsVerNum (OS Version Number) Allowed sortable columns • Name • FQDN • DeviceType • Status • OsType • OsVersionText • AgentVersion • OsVerNum (OS Version Number) • LastSeenTime	
POST	/Consumer/Devices/{includeInherited}	**Minimum API version 8.0**	Requires the same permission

Verb	Request	Notes	Permissions required
		Behaves like the previous API /Consumer/Devices but a new (optional) boolean parameter has been added at the end. If omitted, it defaults to *true*. It is only relevant when the IExpressionTree passed as filter is filtering by one or more Management Groups. When this happens, if the parameter is true, the API returns all the Devices that are members of management groups that inherit from the management group(s) specified in the filter. If the parameter is false, only the direct members of the management group(s) are returned.	s as the API /Consumer/Devices discussed above
POST	/Consumer/Devices/summary	Gets summary information, counting the number of devices grouped by fixed criteria based on static scope. It does not support sorting and pagination. The following example shows a filter that uses all the attributes that	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner

Verb	Request	Notes	Permissions required
		are produced in the output of this API. Under normal circumstances, you would only filter on a subset of these columns. **Example Request** `{ Filter:{ "Operator": "or", "Operands":[ { "Attribute":"OsType", "Operator":"==", "Value":"Windows" }, { "Attribute":"AgentVersion", "Operator":"==", "Value":"281483566841860" }, { "Attribute":"Status", "Operator":"!=", "Value":"2" }, { "Attribute":"DeviceType", "Operator":"==", "Value`	or Approver) on any instruction set

Verb	Request	Notes	Permissions required

		<pre>":"Desktop"
		        },
		        {
		            "Attribute":"TimeZone",
		            "Operator":"==",
		            "Value":"0"
		        },
		        {
		            "Attribute":"VrPlatform",
		            "Operator":"==",
		            "Value":""
		        }
		    ]
		  }
		}</pre> To learn about how to define the filter, check Using scope and filter expressions page.	

POST (FORM)	/Consumer/Devices/exportmatching	Form post endpoint used to return a CSV file containing information about devices. You can include a filter in the form post that contains JSON that conforms scoping and filtering expression as outline above. To do so send JSON as body of the message with a field called "Filter" whose value should be string that	

Verb	Request	Notes	Permissions required
		contains a JSON with the filtering expression. **Example Request Payload** `{ "Operator": "and", "Operands": [ { "Attribute": "DeviceType", "Operator": "==", "Value": "Desktop" }, { "Attribute": "OSType", "Operator": "==", "Value": "Windows" } ] }`	
POST	/Consumer/Devices/ByDomainAndName	Gets device status (Online/Offline) by domain and name. **Example Request Payload** `{ "DomainsAndNames":[ { "Domain":"ACMEFINANCE", "Names":[ "1AF8400000000002", "1AF840`	Requires any of the instruction-related permissions (so Viewer, Questioner, Actioner or Approver)

Verb	Request	Notes	Permissions required

		0000000003"
		]
		},
		{
		"Domain":"ACMEENGINEERING",
		"Names":[
		"1AF8400000000200",
		"1AF8400000000201"
		]
		}
		]
		}
		Example Response Payload
		[
		{
		"Id": 12229,
		"Domain": "ACMEFINANCE",
		"Name": "1AF8400000000003",
		"Status": 0
		},
		{
		"Id": 12305,
		"Domain": "ACMEENGINEERING",
		"Name": "1AF8400000000200",
		"Status": 1
		}
		]
			on any instruction set
			New in Version 8.0: Requires that the invoking user have at least one assignment for the "All Devices" Management Group.

GET	/Consumer/Devices/fqdn/{fqdn}/ManagementGroups	Gets all Management Groups a device belongs to.	Requires any of the instruction-related

Verb	Request	Notes	Permissions required
		FQDN has to be base64 encoded. **Minimum API version 3.3**	permissions (so Viewer, Questioner, Actioner or Approver) onany instruction set
GET	/Consumer/Devices/tachyonguid/{tachyonGuid}/ManagementGroups	Gets all Management Groups a device belongs to. The Guid must be in a format recognizable as a GUID by .NET framework, i.e. 123e4567-e89b-12d3-a456-426655440000 **Minimum API version 4.0**	
GET	/Consumer/Devices/CriticalityMapping	Gets the mapping table for the Criticality of devices. **Minimum API version 4.0** **Example response payload** `{ "0": "Undefined", "1": "Non-critical", "2": "Low", "3": "Medium",`	None.

Verb	Request	Notes	Permissions required
		"4": "High", "5": "Critical" }	
POST	/Consumer/Devices/TachyonGuid/{guid}/Instruction/{forceRefresh}	Sends a Synchronous instruction to the specified device . **Minimum API version 5.2** {guid} should be replaced with the TachyonGuid of the device targeted by the instruction. {forceRefresh} can be set to *true* to bypass the response cache. The default is *false*. The payload for the POST is the same that is used for an ordinary non-synchronous instruction, with the exception that a Device list cannot be provided – the target of the Synchronous instruction should be specified by means of the TachyonGuid parameter.	Requires permissions on the Instruction Set for the selected instruction and on one of the Management Groups to which the device belongs. See Instructions.
GET	/Consumer/Devices/AttributeValues/{attributeName}	Returns a unique list of attribute values for the given attribute. **Minimum API version 8.1**	Requires any of the instruction-related permissions (so

Verb	Request	Notes	Permissions required
		{attributeName} - The name of the attribute for which the values are required. The list of valid attributes is; • OsType • OsVerNum • OsVersionText • AgentVersion • DeviceType • Location	Viewer, Questioner, Actioner or Approver) on any instruction set.

2.3.18 EventSource

Entity - EventSource - handles Event Sources

Minimum API version 5.1

Verb	Request	Notes	Permissions required
GET	/Consumer/EventSource	Returns all event sources	Requires 'Read' permission on 'Event Subscription' securable type
GET	/Consumer/EventSource/{id}	Returns event source by its Id	Requires 'Read' permission on 'Event Subscription'

Verb	Request	Notes	Permissions required
			securable type
POST	/Consumer/EventSource	Adds a new event source	Requires 'Write' permission on 'Event Subscription' securable type
PUT	/Consumer/EventSource	Modifies an existing event source	Requires 'Write' permission on 'Event Subscription' securable type
DELETE	/Consumer/EventSource	Deletes an event source	Requires 'Delete' permission on 'Event Subscription' securable type
POST	/Consumer/EventSource/Search	Returns event sources that fulfil parameters passed in the request body: • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.	Requires 'Read' permission on 'Event Subscription' securable type

Verb	Request	Notes	Permissions required
		• sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Id • Type • Source • LastProcessedTimestampUtc Allowed sort columns: • Id • Type • LastProcessedTimestampUtc	
POST	/Consumer/EventSource/SearchSurveys	Returns event sources that fulfil parameters passed in the request body: • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.	Requires 'Read' permission on 'Event Subscription' securable type

Verb	Request	Notes	Permissions required
		• sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Id • Type • CreatedBy • Source • CreatedTimestampUtc • LastProcessedTimestampUtc • Name • Question • Description • Link • SurveyType • DefaultChoiceIndex • Validation • AllowFreeText • PromptFrequencyDays	

Verb	Request	Notes	Permissions required
		• Deadline • TemplateType Allowed sort columns: • Id • Type • CreatedBy • CreatedTimestampUtc • LastProcessedTimestampUtc • Name • Question • Description • Link • SurveyType • DefaultChoiceIndex • Validation • AllowFreeText • PromptFrequencyDays • Deadline • TemplateType	
POST	/Consumer/EventSource/Shallow/Search	Returns event sources that fulfil parameters passed in the request body. This endpoint returns a subset of information without details about event subscription or management groups.	Requires 'Read' permission on 'Event Subscription'

Verb	Request	Notes	Permissions required
		• filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Id • Type • Source • LastProcessedTimestampUtc Allowed sort columns: • Id • Type • LastProcessedTimestampUtc	securable type
POST	/Consumer/EventSource/Shallow/SearchSurveys	Returns event sources that fulfil parameters passed in the request body. This endpoint returns a subset of information without details	Requires 'Read' permission on 'Event Subscripti

Verb	Request	Notes	Permissions required
		about event subscription or management groups. • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Id • Type • CreatedBy • Source • CreatedTimestampUtc • LastProcessedTimestampUtc • Name • Question • Description • Link	on' securable type

Verb	Request	Notes	Permissions required
		• SurveyType • DefaultChoiceIndex • Validation • AllowFreeText • PromptFrequencyDays • Deadline • TemplateType Allowed sort columns: • Id • Type • CreatedBy • CreatedTimestampUtc • LastProcessedTimestampUtc • Name • Question • Description • Link • SurveyType • DefaultChoiceIndex • Validation • AllowFreeText • PromptFrequencyDays • Deadline • TemplateType	

Verb	Request	Notes	Permissions required
POST	/Consumer/EventSource/SetDirty	Marks the event sources and related data as "Dirty", indicating that a redeploy is needed so that any changes will reach the Clients.	Requires 'Write' permission on 'Event Subscription' securable type

2.3.19 EventSubscriptionAssignments

Entity - EventSubscriptionAssignments - handles Event Subscription Assignments

Minimum API version 5.0

Note for API version 8.0

As of API version 8.0, all the EventSubscriptionAssignmets APIs "hide" from the caller any Management Group on which the caller does not have any Role assignment. This is intended for delegation of administration where the user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results are the ones that you would get if the only existing Management Groups were those on which you have any Role assignment. In other words, trying to access a specific information about event subscription assignments on which you have no rights on an assigned Management Group will return an error, and retrieving information about the event subscription assignment will return only the assigned Management Groups that are accessible to the calling user.

Verb	Request	Notes	Permissions required
POST	/Consumer/EventSubscriptionAssignments	Assigns an EventSubscription to a Management Group	Write on EventSubscription
GET	/Consumer/EventSubscriptionAssignments/{id}	Returns a specific EventSubscription	Read on EventSubscription

Verb	Request	Notes	Permissions required
		assignment object by its Id	
GET	/Consumer/EventSubscriptionAssignments	Returns all event subscription assignments **Example response** `[ { "Id": 4, "SubscriptionId": 2, "ManagementGroupId": 1 }, { "Id": 5, "SubscriptionId": 3, "ManagementGroupId": 1 } ]`	Read on EventSubscription
DELETE	/Consumer/EventSubscriptionAssignments/{id}	Deletes a specific EventSubscription assignment object by its Id	Delete on EventSubscription

2.3.20 EventSubscriptions

Entity - EventSubscriptions - handles Event Subscriptions

Minimum API version 5.0

General note for version 8.0

As of API version 8.0, all the EventSubscriptions APIs that access the assigned Management Groups "hide" from the caller any Management Group on which the caller does not have any Role assignment. This is intended for delegation of administration where the user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results are the ones that you would get if the only existing Management Groups were those on which you have any Role assignment. In other words, trying to access a specific information about event subscriptions on which you have no rights on an assigned Management Group will return an error, and retrieving information about the event subscription will return only the assigned Management Groups that are accessible to the calling user.

Verb	Request	Notes	Permissions required
GET	/Consumer/EventSubscriptions	Returns all Event Subscriptions within the system.	Read on EventSubscription
GET	/Consumer/EventSubscriptions/{id}	Returns a single Event Subscription. Example data returned: **Example response** `{ "Id": 2, "ConsumerId": 2, "Topic": "1E.Performance.*", "Url": "https://target.acme.lab/Test", "Strategy": 0, "Bookmark": null, "Status": 1 }`	Read on EventSubscription

Verb	Request	Notes	Permissions required
POST	/Consumer/EventSubscriptions	Adds an Event Subscription	Write on EventSubscription
PUT	/Consumer/EventSubscriptions	Modifies an existing Event Subscription	Write on EventSubscription
POST	/Consumer/EventSubscriptions/Search	Gets a list of filtered and sorted event subscriptions. **Example Request** `{ "filter":{ "Operator": "AND", "Operands": [ { "Attribute": "ConsumerId", "Operator": "==", "Value": 2 }, { "Attribute": "Status", "Operator": "==", "Value": 1 } ] }, "start": 1, "pageSize":`	Read on EventSubscription

Verb	Request	Notes	Permissions required

		10,
		"sort": [
		{
		"Col
		umn": "Topic",
		"Dir
		ection": "ASC"
		},
		{
		"Col
		umn": "Url",
		"Dir
		ection": "ASC"
		}
		]
		}
		• filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.
		• start - starting index to support pagination. Begins from 1.
		• pagesize - number of results to fetch
		• sort - sort expression. To learn about how	

Verb	Request	Notes	Permissions required
		to define sort criteria check Sort Definition page. Allowed filter columns: • Id • ConsumerId • Topic • Url • Strategy • Status Allowed sortable columns • ConsumerId • Topic • Url • Strategy • Status	
DELETE	/Consumer/EventSubscriptions/{id}	Removes an Event Subscription given its Id	Delete on EventSubscription
POST	/Consumer/EventSubscriptions/{id}/Assignments	Assigns selected Management groups to an Event Subscription	Write on EventSubscription

Verb	Request	Notes	Permissions required
PUT	/Consumer/EventSubscriptions/{id}/Assignments	Replaces all Management groups assigned to the Event Subscription with supplied Management groups	Write on EventSubscription
GET	/Consumer/EventSubscriptions/{id}/Assignments	Returns Management groups assigned to a given Event Subscription	Read on EventSubscription
DELETE	/Consumer/EventSubscriptions/{id}/Assignments	Unassigns specified Management groups from a given Event Subscription	Write on EventSubscription

2.3.21 Expressions

Entity - Expressions - Provides services pertaining expressions

Verb	Request	Notes	Permissions required
POST	/Consumer/Expressions/GetReadable	Converts given expression tree to human readable form	None

POST	/Consumer/Expressions/CheckScope	Checks if given two expressions can be merged into a single valid scope.
		
		Example, two expressions to check wrapped in a JSON object:
		
		Post model
		
		`{`
		`  "First": {`
		`  // 1st scope expression`
		`  "Attribute": "ostype",`
			None

Verb	Request	Notes	Permissions required

		<pre>"Operator": "=",
		"Value": "Windows"
		},
		"Second": {
		// 2nd scope expression
		"Attribute": "rammb",
		"Operator": ">",
		"Value": "2048"
		}
		}</pre> More details can be found on Using scope and filter expressions page.	

2.3.22 Fragments

Entity - Fragments

Minimum API version 4.0

Verb	Request	Notes	Permission required
GET	/Consumer/Fragments	Returns all fragments	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Fragments/{id}	Returns a single fragment by its Id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Fragments/Shallow/All	**Minimum API version 5.1** Returns all fragments but without their respective Payloads.	Requires 'Read' permission on 'GuaranteedSt

Verb	Request	Notes	Permission required
			ate' securable type
POST	/Consumer/Fragments/Search	Returns Fragments that match the search parameters specified Allowed filter columns: • Name • Type • ReadablePayload Allowed sort columns: • Name • Type	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Fragments/Shallow/Search	Returns Fragments that match the search parameters specified. Fragment's payload will not be returned. If you require payload please use /Fragments/Search instead. Allowed filter columns: • Name • Type • ReadablePayload Allowed sort columns: • Name • Type	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permission required
POST	/Consumer/Fragments	**Unavailable from version 5.1 onwards** Creates a new Fragment	Requires 'Write' permission on 'GuaranteedState' securable type
PUT	/Consumer/Fragments	**Unavailable from version 5.1 onwards** Modifies an existing Fragment	Requires 'Write' permission on 'GuaranteedState' securable type
POST	/Consumer/Fragments/UploadFile	**Minimum API version 5.1** Uploads a Fragment file Payload must be sent as multipart form data with the file attached. File must be sent within a boundary. Omitting the boundary will cause the upload to fail. Any boundary can be used as long as it is specified within the call itself. If you wanted to upload an image called 'somefile.xml', you could, for example, use 'xyzzy' as a boundary. Then you would add this line: Content-Type: multipart/form-data; boundary=xyzzy	Requires 'Write' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permission required
		And the payload would look something like this: --xyzzy Content-Disposition: form-data; name="somefile.xml"; filename="somefile.xml" Content-Type: text/xml {the contents of the file goes here} --xyzzy--	
DELETE	/Consumer/Fragments/{id}	Deletes a Fragment	Requires 'Delete' permission on 'GuaranteedState' securable type
GET	/Consumer/Fragments/Export/Id/{id}	**Minimum API version 5.1** Exports a Fragment as an Xml file	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Fragments/Export/Name/{name}	**Minimum API version 5.1** Exports a Fragment as an Xml file	Requires 'Read' permission on 'GuaranteedState' securable type

2.3.23 Information (default)

Entity - Information - Consumer API information (default)

Verb	Request	Notes	Permissions required
GET	/Consumer or /Consumer/Information	This endpoint returns Consumer API information which includes version. **Example Response** `{ "Version": "1.0.0.0" }`	None

2.3.24 InstructionDefinitions

Entity - Instruction Definitions - Interrogates the InstructionDefinition table

Verb	Request	Notes	Permissions required
GET - all	/Consumer/InstructionDefinitions?instructionType={type} or /Consumer/InstructionDefinitions (to get all the instruction definitions assigned to the user)	Gets a list of instruction definitions of specified type(s). Instruction types can be • Question • Action More than one type can be asked for. For example, to get both Action and Question types /Consumer/InstructionDefinitions?instructionType=Action&instructionType=Question **If no type is specified, all instruction definitions are returned.** ***Only instructions that the user can see as per their permissions are returned.***	Questions • Questioner • Actioner Actions • Actioner It does not return Unauthorized (401) rather filters out instruction definitions based on

Verb	Request	Notes	Permissions required
			permissions.
GET - all	/Consumer/InstructionDefinitions/Parent/{parentId}	**Minimum API version 3.3.** Gets a list of instruction definition that can be issued as a follow-up to specific instruction. This endpoint will take into account permissions of the parent instruction and calling user's permission to filter out instruction definitions that wouldn't be sent to any devices due to restrictions imposed by management groups. parentId should be a valid instruction id (not an instruction definition id). Instruction types can be • Question • Action More than one type can be asked for. For example, to get both Action and Question types /Consumer/InstructionDefinitions?instructionType=Action&instructionType=Question **If no type is specified, all instruction definitions are returned.** ***Only instructions that the user can see as per their permissions are returned.***	Questions • Questioner • Actioner Actions • Actioner It does not return Unauthorized (401) rather filters out instruction definitions based on permissions.

GET - si	/Consumer/InstructionDefinitions/Id/{id}	Gets an instruction definition by ID.
		
		Example Response
		
		`{`
		`  "Id": 21,`
		`  "Name": "Disabled services",`
			Questions
			• Questi

Verb	Request	Notes	Permissions required
ngle		"Description": "Shows count of disabled services", "ProductPackId": 2, "InstructionType": 0, "ReadablePayload": "Show count of disabled services", "Parameters": null, "Schema": [{ "Name": "Caption", "Type": "string", "Length": 256 }], "Aggregation": { "Schema": [{ "Name": "Caption", "Type": "string", "Length": 256, "RenderAs": "preformatted" }, { "Name": "Count", "Type": "int32", "Length": 0 }], "GroupBy": "Caption", "Operations": [{ "Name": "Count", "Type": "count" }] }, "Category": "Operating System", "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 120, "MinimumInstructionTtlMinutes": 10, "MaximumInstructionTtlMinutes": 1440, "MinimumResponseTtlMinutes": 10, "MaximumResponseTtlMinutes":	one r • Acti one r Actions • Acti one r Returns Unauthorized (401) if user does not have the required permissions at global level or on the instruction definition's product pack.

Verb	Request	Notes	Permissions required
		10080, "IsLicensed" = true, "NumberOfTimesExecuted": 7 }	
GET - single by name	/Consumer/InstructionDefinitions/Name/{name}	Gets an instruction definition by name.	Questions • Questioner • Actioner Actions • Actioner Returns Unauthorized (401) if user does not have the required permissions at global level or on the instruction definition's product pack.
GET - list	/Consumer/InstructionDefinitions/Search/{searchString}?instructionType={type}	Gets a list of instruction definitions of specified type matching search terms in description and readable payload. For example:	Questions • Questi

Verb	Request	Notes	Permissions required
by search terms		/Consumer/InstructionDefinitions/Search/windows network?type=Question It will return instruction definitions of Question type matching search terms "windows" and "network" More than one type can be mentioned in the query string. **If no type is mentioned, all instruction definitions matching search terms are returned regardless of their types.** **API versions 3.1 and later: the searchString must be encoded as Base64.** This was needed because otherwise the server threw errors whenever the search string contained some special characters such as a dot. The API understands both strict Base64 and URL-safe Base64. It is recommended that you use the latter, since otherwise there are some possible search strings that would result in malformed URLs when the Base64-encoded text is appended. For backwards compatibility, if the version specified in the API call is earlier than 3.1, the API expects the search string to NOT be encoded in Base64.	one r • Actione r Actions • Actione r It does not return Unauthorized (401) rather filters out instruction definitions based on permissions.
GET-list by sear	/Consumer/InstructionDefinitions/Parent/{parentId}/Search/{base64SearchString}?instructionType={type}	**Minimum API version 3.3** Gets a list of instruction definitions of specified type matching search terms in description and readable payload that can be issued as a follow up to given instruction. This endpoint will take into account permissions of the	Questions • Questione r

Verb	Request	Notes	Permissions required
chterms		parent instruction and calling user's permission to filter out instruction definitions that wouldn't be sent to any devices due to restrictions imposed by management groups. parentId should be a valid instruction id (not an instruction definition id). More than one type can be mentioned in the query string. **If no type is mentioned, all instruction definitions matching search terms are returned regardless of their types.** The API understands both strict Base64 and URL-safe Base64. It is recommended that you use the latter, since otherwise there are some possible search strings that would result in malformed URLs when the Base64-encoded text is appended.	• Actioner Actions • Actioner It does not return Unauthorized (401) rather filters out instruction definitions based on permissions.
POST	/Consumer/InstructionDefinitions/Search	**Minimum API version 4.0** This endpoint allows searching for instruction definitions based on criteria along with sorting and pagination. This endpoint is designed to be used when managing instruction definitions and instruction sets, not when issuing instructions because it will return all instruction definitions in the system, not just the ones calling user has permissions on. **Example request payload** `{ "filter":{ "Operator": "and",`	Required 'Read' permission on 'InstructionSetManagement'

Verb	Request	Notes	Permissions required

		<pre>"Operands":[
		{
		"Attribute":"Name",
		"Operator":"LIKE",
		"Value":"What%"
		},
		{
		"Attribute":"Licensed",
		"Operator":"==",
		"Value":"true"
		}
		]
		},
		"start":10,
		"pageSize":100,
		"sort":[
		{
		"Column":"Name",
		"Direction":"ASC"
		}
		]
		}</pre>	

DELETE	/Consumer/InstructionDefinitions/Name/{instructionName}	**Minimum API version - 3.1** Removes instruction definition with a given name. Will fail if there are pending instructions based on that definition	Requires 'Delete' permission on Instruction set management
DELETE	/Consumer/InstructionDefinitions/Id/{instructionId}	**Minimum API version - 3.1** Removes instruction definition with a given id. Will fail if there are pending instructions based on that definition	Requires 'Delete' permission on Instruction set

Verb	Request	Notes	Permissions required
			management
DELETE	/Consumer/InstructionDefinitions	**Minimum API version - 3.1** Deletes multiple instruction definitions defined by name or Id. You cannot use a mix of names and Id. Has to be one or the other. **Payload JSON** `{ "Ids": [], "Names": [] }` Provide a JSON with either 'Ids' or 'Names' array filled in	Requires 'Delete' permission on Instruction set management
GET	/Consumer/InstructionDefinitions/InstructionSet/Name/{setName}	**Minimum API version - 3.1** Returns instruction definitions from a specific set by set name	Requires 'Read' permission on instruction set management
GET	/Consumer/InstructionDefinitions/InstructionSet	**Minimum API version - 3.1** Returns all instruction definitions that are assigned to an instruction set	Requires 'Read' permission on instruction set management
GET	/Consumer/InstructionDefinitions/All	**Minimum API version - 3.1**	Requires 'Read' permission on

Verb	Request	Notes	Permissions required
		Returns all instruction definitions. Includes both assigned and unassigned instruction definitions	instruction set management
GET	/Consumer/InstructionDefinitions/InstructionSet/Id/{setId?}	**Minimum API version - 3.1** Returns instruction definitions from a specific set by set id. If no Id is provided all unassigned instruction definitions are returned.	Requires 'Read' permission on instruction set management
POST	/Consumer/InstructionDefinitions/Id/{defId:int}/InstructionSet/{setId:int}	**Minimum API version - 3.1** Adds instruction definition with given Id to an instruction set with a specific Id	Requires 'Add' permission on instruction set management
DELETE	/Consumer/InstructionDefinitions/Id/{defId:int}/InstructionSet/{setId:int}	**Minimum API version - 3.1** Removes instruction definition with given Id from an instruction set with a specific Id	Requires 'Delete' permission on instruction set management
GET	/Consumer/InstructionDefinitions/Name/{name}/export	**Minimum API version - 3.1** Exports instruction definition with a given name to Xml that can be downloaded from the API as an Xml file (MIME application/xml)	Requires 'Read' permission on instruction set management

Verb	Request	Notes	Permissions required
GET	/Consumer/InstructionDefinitions/Id/{id:int}/export	**Minimum API version - 3.1** Exports instruction definition with a given Id to Xml that can be downloaded from the API as an Xml file (MIME application/xml)	Requires 'Read' permission on instruction set management
POST	/Consumer/InstructionDefinitions/Export	**Minimum API version - 3.2** Exports a collection of instruction definitions into a ZIP archive. **Request payload** `{ "Ids": [], "Names": [] }` You can provide an array of instruction definition Ids or names, but not both. All Ids or names must be valid.	Requires 'Read' permission on instruction set management
GET	/Consumer/InstructionDefinitions/ManagementGroup/{id:int?} **From 5.0 onwards:** /Consumer/InstructionDefinitions/ManagementGroup/{id:int?}?includeAllDevices=true/false	**Minimum API version - 3.3** Returns Management Groups calling user is allowed to access. 'All Devices' management group will not included in the results even if calling user has permission to it. See below for details. Optionally, an Instruction Definition Id can be supplied. If it is supplied, only Management Groups accessible for that Instruction Definition will be returned. If it is not supplied, all Management Groups will be returned regardless of	This endpoint does not require specific permissions but will return an empty array if the user has no permissions to access Management Groups.

Verb	Request	Notes	Permissions required
		which Instruction Definition they are accessible for. From version **5.0** onwards this endpoint supports an optional query string parameter called includeAllDevices. If this parameter is supplied and its value is 'true', 'All Devices' management groups will be returned in the search results (obviously if calling user can access it). Omitting this parameter or setting any other value will result in this management group being removed from the search results. **New for API version 8.0**: When returning the Management Groups, the API also includes those that are inferred from the Management Group inheritance hierarchy, in addition to those directly assigned to the specified instruction definition.	
GET	/Consumer/InstructionDefinitions/ManagementGroup/Search/{base64SearchString}	**Minimum API version - 3.3** Returns Management Groups calling user is allowed to access whose Names are at least a partial match for specified search string. 'base64SearchString' must be the base64 encoded search string. All Management Groups will be returned regardless of which Instruction Definition they are accessible for. **New for API version 8.0**: When returning the Management Groups,	This endpoint does not require specific permissions but will return an empty array if the user has no permissions to access

Verb	Request	Notes	Permissions required
		the API also includes those that are inferred from the Management Group inheritance hierarchy, in addition to those directly assigned to the specified instruction definition.	Management Groups.
GET	/Consumer/InstructionDefinitions/ManagementGroup/{id:int}/Search/{base64SearchString}	**Minimum API version - 3.3** Returns Management Groups calling user is allowed to access whose Names are at least a partial match for specified search string. 'base64SearchString' must be the base64 encoded search string. 'id' is the Id of an Instruction Definition in whose context you want to get the Management Groups Only Management Groups accessible for that Instruction Definition will be returned. **New for API version 8.0**: When returning the Management Groups, the API also includes those that are inferred from the Management Group inheritance hierarchy, in addition to those directly assigned to the specified instruction definition.	This endpoint does not require specific permissions but will return an empty array if the user has no permissions to access Management Groups.
GET - list by product pa	/Consumer/InstructionDefinitions/ProductPackId/{productPackId}	***REMOVED IN 3.1*** Gets all instruction definitions belonging to a product pack by its id.	• Read - ProductPack Management

Verb	Request	Notes	Permissions required
ck id			Returns Unauthorized (401) if user does not have the required permission.
GET - list by product pack name	/Consumer/InstructionDefinitions/ProductPackName/{productPackName}	***REMOVED IN 3.1*** Gets all instruction definitions belonging to a product pack by its name.	• Read - ProductPack Management Returns Unauthorized (401) if user does not have the required permission.

2.3.25 InstructionHierarchies

Entity - InstructionHierarchies - Interrogates related instruction definitions

Verb	Request	Notes	Permissions required
GET	/Consumer/InstructionHierarchies/{id}	Returns the instruction hierarchy for a given instruction Id.	None

Verb	Request	Notes	Permissions required
		Follow-up instructions will be listed under the "Children" node Parent instructions will be listed under the "UpperLevels" node The last entry in the "UpperLevels" node will be the instruction itself **New for API version 8.0**: An error will be returned if the specified instruction is not accessible to the invoking user as per the Management Group criteria as stated in the note at the top of the page Issue an instruction, track its progress and retrieve responses. Note that the call will return all the instructions in the Hierarchy, regardless of Management Group permissions. Under normal circumstances, the hierarchy will not contain any disallowed instructions, because the Management Group permissions check would not have allowed to send the child without permissions on the parent. However, if the Management Group assignments of the users are changed after the instructions were sent, there is a chance that the hierarchy might contain a a disallowed instruction. If the user then attempts to access such instruction, an error will be returned.	

2.3.26 Instructions

Entity - Instructions (Questions and Actions) - Interrogates the Instruction table

Note for API version 8.0

As of API version 8.0, Tachyon implements changes in the accessibility of instructions and their responses with the goal of enabling delegation of administration. The user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. From the point of view of the Instructions API, this implies that the user is not allowed to see any instructions whose set of results might potentially contain responses from endpoints that should be invisible to the user. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results hide any instructions not accessible to the invoking user.

To this end, the system considers "visible to the user" the instructions that were submitted by any other user whose Management Group assignments are a subset of the Management Group assignments of the user that invokes the API. For this purpose, the system uses the **current** Management Groups assigned to each user, **not** the Management Groups that the users could access back at the time when the instruction was submitted.

Verb	Request	Notes	Permissions required
GET - single	/Consumer/Instructions/{instructionId}	Gets an individual question or action by an Id. **Example Response** Expand source `{ "Id": 1010, "Sequence": 19, "Name": "1E-Explorer-TachyonCore-BiosDetails", "Description": "Returns details of BIOS firmware.", "InstructionType": 0, "ReadablePayload": "What BIOS firmware is installed?", "Cmd": "SendAll",`	• Viewer • Questioner • Actioner • Approver Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDef

Verb	Request	Notes	Permissions required

		<pre>"Schema": [
		{
		"Name": "BIOSVersion",
		"Type": "string",
		"Length": 512,
		"RenderAs": null
		},
		{
		"Name": "Caption",
		"Type": "string",
		"Length": 512,
		"RenderAs": null
		},
		{
		"Name": "Manufacturer",
		"Type": "string",
		"Length": 512,
		"RenderAs": null
		},
		{
		"Name": "PrimaryBIOS",
		"Type": "bool",
		"Length": 0,
		"RenderAs": null
		},
		{
		"Name": "ReleaseDate",
		"Type": "datetime",
		"Length":</pre>
			inition's product pack.

Verb	Request	Notes	Permissions required


```
0,
                "RenderAs
": null
            },
            {
                "Name": "S
erialNumber",
                "Type": "s
tring",
                "Length":
512,
                "RenderAs
": null
            }
        ],
        "Aggregation": {
            "Schema": [
                {
                    "Name
": "Manufacturer",
                    "Type
": "string",
                    "Lengt
h": 512,
                    "Rende
rAs": null
                },
                {
                    "Name
": "BIOSVersion",
                    "Type
": "string",
                    "Lengt
h": 512,
                    "Rende
rAs": null
                },
                {
                    "Name
": "Count",
                    "Type
": "int64",
                    "Lengt
h": 0,
                    "Rende
rAs": null
                }
```

Verb	Request	Notes	Permissions required


```
        ],
        "GroupBy": "Manufacturer,BIOSVersion",
        "Operations": [
            {
                "Name": "Count",
                "Type": "count"
            }
        ]
    },
    "KeepRaw": true,
    "Scope": {
        "Attribute": "OsType",
        "Operator": "==",
        "Value": "Windows",
        "DataType": "String"
    },
    "ReadableScope": {
        "Attribute": "OsType",
        "Operator": "==",
        "Value": "Windows",
        "DataType": "String"
    },
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "CreatedTimestampUtc": "2018-07-13T08:27:05.7Z",
    "SentTimestampUtc": "2018-07-13T08:27:37.323Z",
    "Status": 6,
    "WorkflowState":
```

Verb	Request	Notes	Permissions required

		5,
		"StatusTimestampUtc": "2018-07-13T09:28:04.61Z",
		"Export": false,
		"ExportLocation": "",
		"ParentInstructionId": null,
		"InstructionDefinitionId": 85,
		"CreatedBy": "SomeDomain\\Some.User",
		"ResultsFilter": null,
		"PreviousResultsFilter": null,
		"ConsumerId": 1,
		"ConsumerName": "Explorer",
		"ConsumerCustomData": null,
		"ParameterJson": null,
		"OffloadResponses": false,
		"RequestedFor": null,
		"ResponseTemplateId": 1,
		"ResponseTemplateConfiguration": {
		"Name": "default",
		"TemplateConfigurations": [
		{
		"Id": "mainchart",
		"Title": "Most common BIOS manufacturers and versions",
		"Type": "Bar",
		"X": "BIOSVersion",	

Verb	Request	Notes	Permissions required

		"Y": "Count",
		"Z": "Manufacturer",
		"PostProcessor": "processingFunction",
		"Size": 1,
		"Row": 1
		}
		],
		"PostProcessors": [
		{
		"Name": "processingFunction",
		"Function": "ProcessMultiSeries('BIOSVersion', 'Count', 'Manufacturer', '5', '5', 'false')"
		}
		]
		},
		"Workflow": "{\"StateMachine\":\"State\"}",
		"Comments": null,
		"ScheduledInstructionId": null,
		"ApprovalOffloaded": false,
		"ActionedBy": null,
		"ActionReason": null
		}	

POST	/Consumer/Instructions/search	To search instructions based on criteria along with sorting and pagination.	• Viewer • Questioner

Verb	Request	Notes	Permissions required

		Example Request
		
		{
		"filter":{
		"Operator": "AND",
		"Operands": [
		{
		"Attribute": "Status",
		"Operator": "==",
		"Value": "5"
		},
		{
		"Attribute": "CreatedTime",
		"Operator": ">=",
		"Value": "2016-04-21T06:30:00Z"
		},
		{
		"Attribute": "CreatedTime",
		"Operator": "<=",
		"Value": "2016-04-30T19:55:00"
		}
		]
		},
		"start": 1,
		"pageSize": 10,
		"sort": [
		{
		"Column": "payload",
		"Direction": "ASC"
		},
		{
		"Column": "CreatedTime",
			• Actioner
			• Approver
			
			It does not return Unauthorized (401) rather filters out instructions based on permissions.

Verb	Request	Notes	Permissions required

		<pre>"Direction": "DESC"
		}
		]
		}</pre>
		• filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.
		• start - starting index to support pagination. Begins from 1.
		• pagesize - number of results to fetch
		• sort - sort expression. To learn about how to define sort criteria check Sort Definition page.
		Allowed filter columns:
		• ReadablePayload
		• Description
		• CreatedTime
		• Status
		• DefinitionName (v1.6)
		• DefinitionId (v1.6)
		• ConsumerId (2.0)	

Verb	Request	Notes	Permissions required
		• ConsumerCustomData (v1.6) Allowed sortable columns • ReadablePayload • Description • CreatedTime • Status **NOTE:** Only system consumer can search for other consumers' instructions. Other consumers will only see their instructions.	
POST	/Consumer/Instructions	Creates a question or action. **Example Request** `{ "DefinitionId": 9, "DefinitionName": null, "Scope": { "Attribute": "OsType", "Operator" : "=", "Value": "Windows" }, "Parameters": null, "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 120, "ParentInstructionId": null,`	For questions • Questioner • Actioner For actions • Actioner Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack.

Verb	Request	Notes	Permissions required


```
    "KeepRaw": false,
    "Export": false,
    "ExportLocation":
null,
    "ResultsFilter": {
                "Attri
bute": "NumberOfCores
",
                "Opera
tor": ">",
                "Value
": "2"
        },
        "PreviousResul
tsFilter": null,
    "ConsumerCustomDat
a": "MyCustomData",
    "OffloadResponses
": false
}
```


will return:

Example Response


```
{
        "Id": 199,
        "Name": "Syste
m Processor Details",
        "Description":
"Represents a device c
apable of interpreting
a
sequence of machine in
structions on a comput
er system running Wind
ows",
        "InstructionTy
pe": 0,
        "ReadablePaylo
ad": "Get Processor De
tails",
        "Cmd": "SendAl
l",
        "Schema": [
            {
                "Name
```

Verb	Request	Notes	Permissions required


```
": "Caption",
                "Type
": "string",
                "Lengt
h": 256
            },
            {
                "Name
": "Description",
                "Type
": "string",
                "Lengt
h": 256
            },
            {
                "Name
": "Manufacturer",
                "Type
": "string",
                "Lengt
h": 256
            },
            {
                "Name
": "NumberOfCores",
                "Type
": "int32",
                "Lengt
h": 0
            }
        ],
            "Aggregation":
null,
            "KeepRaw": fal
se,
            "Scope": null,
            "QuestionTtlMi
nutes": 60,
            "AnswerTtlMinu
tes": 120,
            "CreatedTimest
ampUtc": "2015-10-01T0
8:36:38.663",
            "Status": 0,
            "AnswerCount":
0,
            "SentCount":
```

Verb	Request	Notes	Permissions required

		<pre>0,
		"ParentQuestionId": null,
		"QuestionDefinitionId": 9,
		"CreatedBy": "",
		"ResultsFilter": {
		"Attribute": "NumberOfCores",
		"Operator": ">",
		"Value": "2"
		},
		"PreviousResultsFilter": null,
		"ConsumerId": 0,
		"ConsumerCustomData": "MyCustomData",
		"OffloadResponses": false,
		"ParameterJson" : null
		}</pre>	

POST	/Consumer/Instructions/Targeted	Creates a question or action, supplying a targeted list of devices by Fqdn
		
		As above with the added "Devices" Json property e.g.
		
		Example Request
		
		<pre>{
		"DefinitionId": 9,
		"DefinitionName": null,
		"Parameters": null,
		"InstructionTtlMin</pre>
			As above
			
			Note: Fqdn list is limited by MaxRequestLength which defaults to 4MB

Verb	Request	Notes	Permissions required

		<pre>utes": 60,
		"ResponseTtlMinutes": 120,
		"ParentInstructionId": null,
		"KeepRaw": false,
		"Export": false,
		"ExportLocation": null,
		"ResultsFilter": {
		"Attribute": "NumberOfCores",
		"Operator": ">",
		"Value": "2"
		},
		"PreviousResultsFilter": null,
		"ConsumerCustomData": "MyCustomData",
		"OffloadResponses": false,
		"Devices":["pedvwks127.pe.local","pedvwks126.pe.local","pedvwks240.pe.local"]
		}</pre>	

POST	/Consumer/Instructions/{instructionId}/cancel/{keepData}	Cancels an instruction. • id - question to stop • keepData - flag to indicate if result should be deleted	For questions • Questioner • Actioner For actions • Actioner Returns Unauthorized (401) if user

Verb	Request	Notes	Permissions required
			does not have the required permissions at global level or on the parent InstructionDefinition's product pack. Instruction's owner can also cancel.
POST	/Consumer/Instructions/{instructionId}/rerun	Reruns a question. Rerunning an action is not allowed, as it can be risky.	For questions • Questioner • Actioner Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack.
GET	/Consumer/Instructions/{instructionId}/targetlist	Gets targets of given instruction	For questions • Questioner • Actioner Returns Unauthorized (401) if user

Verb	Request	Notes	Permissions required
			does not have the required permissions at global level or on the parent InstructionDefinition's product pack.
GET	/Consumer/Instructions/InFlight/InstructionSet/{instructionSetId}	Gets "In-Flight" instructions for a given Instruction Set. The result has this shape: **Example Request** `{ "TotalCount": 2, "Items": [ { "Id": 2401, "Name": "TachyonCore-BiosDetails", ….. } ] }` The "TotalCount" property contains the **total available In-flight instructions**. The "Items" property contains the **list of In-flight instructions which the user can see**. (and therefore its length could be less than TotalCount).	Filters out the results based on permissions.
GET	/Consumer/Instructions/{instructionId}/proxtarget	**Minimum API version - 3.3**	Requires any permission on any of the

Verb	Request	Notes	Permissions required
		Gets approximate target for specific instruction. Approximate target will be calculated using the scope defined in the instruction. Approximate target will be calculated for the user who created the instruction, using that user's current permissions. **New for API version 8.0**: The calculation for the approximate target only counts the Devices that the current user is allowed to see in accordance to the Management Groups that are assigned to the user.	instruction sets

2.3.27 InstructionSets

Entity - InstructionSets - handles instruction set management

Minimum API version 3.1

Verb	Request	Notes	Permissions required
GET	/Consumer/InstructionSets/	Returns all instruction sets **Return payload** `[{ "Id" : 1 "Name" : "MyFirstGroup", "Description" : "somegroup", "Icon" :`	'Read' on instruction set management

Verb	Request	Notes	Permissions required

		<pre>[]
		},
		{
		"Id" : 2
		"Name" : "MyOtherGroup",
		"Description" : "someothergroup",
		"Icon" : []
		}]</pre>	

GET	/Consumer/InstructionSets/IncludeCounts	Returns all instruction sets and also includes the number of instructions assigned to each set
		
		Return payload
		
		<pre>[{
		"Id" : 1
		"Name" : "MyFirstGroup",
		"Description" : "somegroup",
		"Icon" : [],
		"CountOfDefinitions": 194
		},
		{
		"Id" : 2
		"Name" : "MyOtherGroup",
		"Description" : "someothergroup",
		"Icon" : [],
		"CountOf</pre>
			'Read' on instruction set management

Verb	Request	Notes	Permissions required

		<pre>Definitions": 0
		},
		{
		"Id": null,
		"Name": "Unassigned",
		"Description": "Instruction definitions not assigned to a set",
		"Icon": null,
		"CountOfDefinitions": 9
		}]</pre> Note the last entry, Id=null, which reports the total number of instruction definitions that have not been assigned to any set.	

GET	/Consumer/InstructionSets/Id/{id}	Returns instruction set by Id
		Return payload
		<pre>{
		"Id" : 1
		"Name" : "MyFirstGroup",
		"Description" : "This is my first group",
		"Icon" : []
		}</pre>
			'Read' on instruction set management

Verb	Request	Notes	Permissions required
GET	/Consumer/InstructionSets/Name/{name}	Returns instruction set by name Payload as above	'Read' on instruction set management

POST	/Consumer/InstructionSets/	Creates a new instruction set
		
		Payload
		
		{
		"Name" : "",
		"Description" : "",
		"Icon" : []
		}
		
		Note - icon is in binary form
			'Add' on instruction set management

PUT	/Consumer/InstructionSets/	Modifies an existing instruction set Payload same as for the POST method above	'Add' on instruction set management
DELETE	/Consumer/InstructionSets/Id/{id}?deleteContent={true/false}	Deletes an instruction set with a specific Id. All instruction definitions that belong to that set will become 'unassigned' unless you specify 'true'	'Delete' on instruction set management

Verb	Request	Notes	Permissions required
		on the optional deleteContent flag.. OPTIONAL - you can add deleteContent parameter to the query string. If this parameter is set to true all instruction definitions that belong to the set being deleted will be deleted as well. NOTE - This operation will fail if any of the instruction definition being affected have pending instructions.	
GET	/Consumer/InstructionSets/Id/{id}/Export	Exports all instructions in a given set to a zip file which will contain each definition in a separate xml file. This Zip file can be uploaded as as product pack	'Read' on instruction set management
POST	/Consumer/InstructionSets/Contents	Adds given instruction definitions to a specific instruction set	'Add' on instruction set management

Verb	Request	Notes	Permissions required
		Payload `{ "SetId": 0, "InstructionDefinitionIds" : [0,1,2,...] }` SetId can be a valid Instruction Set Id or null, which means 'unassigned'. InstructionDefinitionIds array had to contain valid instruction definition Ids. None of the instruction definitions can have in-flight instructions.	
DELETE	/Consumer/InstructionSets/Contents	Removes given instruction definitions from a specific instruction set. Payload is the same as in the Add method above. SetId has to be a valid Instruction Set Id.	'Delete' on instruction set management
DELETE	/Consumer/InstructionSets/Id/{id}/Clear	Removes all instruction	'Read' on instruction set

Verb	Request	Notes	Permissions required
		definitions from given set. This does not delete the set itself or any of the instruction definitions. The set becomes empty and all instruction definitions become 'unassigned' NOTE - This operation will fail if any of the instruction definition being affected have pending instructions.	management

2.3.28 InstructionStatistics

Entity - Instruction Statistics - handles instruction statistics

Note for API version 8.0

As of API version 8.0, the visibility of the instructions is restricted in accordance to the Management Groups assigned to the users. See the note at the top of the Instructions page for an explanation of how this works. The same principle applies to the InstructionStatistics API, where it is not possible to access the statistics for an instruction if the instruction itself is not accessible.

Verb	Request	Notes	Permissions required
GET	/Consumer/InstructionStatistics	Returns statistics for all instructions	It does not return Unauthorized (401) rather filters out

Verb	Request	Notes	Permissions required
			instructions based on permissions.
GET	/Consumer/InstructionStatistics/{id}	**OBSOLETE from 3.3 (INC) onwards. Use /Consumer/InstructionStatistics/Summary/{id} instead** Returns statistics for specific instruction	For questions • Viewer • Questioner • Actioner • Approver Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent Instruction Definition's product pack.
GET	/Consumer/InstructionStatistics/Summary/{id}	Returns summarized statistics for specific instruction	For questions

Verb	Request	Notes	Permissions required
			• Viewer • Questioner • Actioner • Approver Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent Instruction Definition's product pack.
GET	/Consumer/InstructionStatistics/Detail/{id}	Returns detailed instruction statistics for specific instruction	For questions • Viewer • Questioner

Verb	Request	Notes	Permissions required
			• Actioner • Approver Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent Instruction Definition's product pack.
GET	/Consumer/InstructionStatistics/Combined/{id}	Returns summarized and detailed statistics for specific instruction	For questions • Viewer • Questioner • Actioner • Approver

Verb	Request	Notes	Permissions required
			Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent Instruction Definition's product pack.
POST	/Consumer/InstructionStatistics/Multiple	**Minimum API version 8.0** Retrieves Statistics for multiple instructions. Input: list of instruction IDs, for example: [10, 45,18] Output: Array of InstructionStatistics objects.	The API is open to any caller, but the results returned will be filtered in accordance with the permissions and Management Groups of the caller.

2.3.29 ManagementGroups

Entity - ManagementGroups - handles management groups

Minimum API version 3.3

Note for API version 8.0

As of API version 8.0, access to management groups is restricted by the Management Group assignments of the user that makes the request. Other than the exceptions noted

below, this means that these APIs will return an error if the user explicitly requests information about a Management Group that he or she is not allowed to use, and the APIs that return a list of Management Groups will exclude from the list any Management Groups that are not allowed for the caller.

Deprecation of Internal mode

As of API version 8.0, the "Internal" mode for creating, updating and deleting Management Groups has been deprecated. The Platform (SLA) APIs have been extended to allow similar functionality as was previously provided by the corresponding ManagementGroup Consumer APIs, so any future development that requires individual manipulation of Management Groups is expected to be done through SLA rather than by invoking the Consumer API.

The obsolete Consumer APIs are those marked below as "*Requires Tachyon Management Group support mode to be set to 'Internal'*." For backwards compatibility, these APIs can still be invoked in the v8.0 release , but they have been removed from the Swagger documentation and have been marked as Obsolete in the Consumer SDK. You should avoid using them for any new software development, as they will be permanently removed in future versions.

Verb	Request	Notes	Permissions required
POST	/Consumer/ManagementGroups/InitiateSlaSync	Initiate synchronization of Management Groups with SLA. Requires Tachyon Management Group support mode to be set to 'External'. **Payload** `{ 'Repository':'Default Inventory' }` Change 'Default Inventory' into the name of the SLA repository against which you are synchronizing. It needs to match the repository that is configured in the Coordinator, otherwise you will get back an error.	Requires 'Synchronize' permission on Management Groups
GET	/Consumer/ManagementGroups	Returns all Management Groups. By default, only non-system Management Groups are returned. In order to get system groups as well, use includeSystemGroups query	Requires one of following:

Verb	Request	Notes	Permissions required
		string parameter and set it to true i.e. https://your.tachyon.server.address/Consumer/ManagementGroups?includeSystemGroups=true **Return payload** `[{ "Id": "1", "Name": "Management Group A", "Description": "", "Expression": "", "Count": "12", "UsableId": "0E9514B6-FD77-4077-9B54-A2AAA0C1953C", "HashOfMembers": "5c9f347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5b754" },{ "Id": "2", "Name": "Management Group B", "Description": "", "Expression": "", "Count": "7", "UsableId": "AD9514B6-FD77-4077-9B54-A2AAA0C1953C", "HashOfMembers": "ddff347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5bbcc" }]` **New as of API version 8.0** : This API will return all the Management Group that the calling user can see regardless of context. Only useful for Device list; in all other instances callers should use ManagementGroups/SecurableType/{securableType}/Operation/{operation} By default this endpoint will return a flat list of Management Groups like it did in previous	• Any permission on any of the InstructionSets • 'Read' perm

Verb	Request	Notes	Permissions required
		versions. As of version 8.0 this endpoint will accept an optional query string parameter called **view**. If that parameter is set to **tree**, this endpoint will return the Management Groups as a tree structure.	isi on on Ma na ge me nt Gr ou ps

GET	/Consumer/ManagementGroups/Id/{id}	Returns a single Management Group by its Id.
		Return payload
		<pre>{
		"Id": "1",
		"Name": "Management Group A",
		"Description": "",
		"Expression": "",
		"Count": "12",
		"UsableId": "0E9514B6-FD77-4077-9B54-A2AAA0C1953C",
		"HashOfMembers": "5c9f347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5b3ea"
		}</pre>
			Requires one of following:
			• Any permission no

Verb	Request	Notes	Permissions required
			any of the InstructionSets • 'Read' permission on ManageE

Verb	Request	Notes	Permissions required
			mentGroups
GET	/Consumer/ManagementGroups/Name/{name}	Returns a single Management Group by its Name. Name needs to be supplied as Base64Encoded string. Return payload as above.	Requires one of following: • Any permission on any of the Instr

Verb	Request	Notes	Permissions required
			uctionSets • 'Read' permission on ManagementGroups

Verb	Request	Notes	Permissions required
GET	/Consumer/ManagementGroups/UsableId/{usableId}	Returns a single Management Group by its UsableId. Return payload as above.	Requires one of following: • Any permission on any of the InstructionSets • 'Rea

Verb	Request	Notes	Permissions required
			d' p e r m is si o n o n M a n a g e m e n t G r o u p s
GET	/Consumer/ManagementGroups/Search/{base64SearchString}	Returns a collection of Management Groups whose names are at least a partial match to the search string. System Management Groups are not returned. Search string must be Base64 encoded. Return payload will be as in GET /Consumer/ManagementGroups. **New as of API version 8.0** : This API will search all the Management Groups that the calling	Requires one of following : • A n y p e r m

Verb	Request	Notes	Permissions required
		user can see regardless of context. Only useful for Device list, in all other instances callers should use ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Search/{base64SearchString}	is si o n o n a n y o f t h e In st r u ct io n S e ts • 'R e a d' p e r m is si o n o n M

Verb	Request	Notes	Permissions required
			anagementGroups
POST	/Consumer/ManagementGroups/Search	Returns Management Groups that match search parameters passed in the request body: • filter - filter expression.To learn about how to define filter checkUsing scope and filter expressionspage. • sort - sort expression. To learn about how to define sort criteria checkSort Definitionpage. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Name • CreatedTime • ModifiedTime • Description	Requires one of following: • Any permission on any of t

Verb	Request	Notes	Permissions required
		• UsableId Allowed sort columns: • Name • CreatedTime • ModifiedTime Example **POST payload** `{ "Filter": { "Operator": "and", "Operands":[ { "Attribute":"Name", "Operator": "like", "Value":"Group%" }, { "Attribute":"UsableId", "Operator": "like", "Value":"%2" } ] }, "Start": 100, "PageSize": 100, "Sort": [{Column: "CreatedTime", Direction: "DESC"}] }` **New as of API version 8.0** : Will search all Management Groups that the calling user can see regardless of context. Only useful for Device list, in all other instances callers should use	• 'Read' permission on Management Group • 'Read' permission on Instruction Sets

Verb	Request	Notes	Permissions required
		ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Search	ups
POST	/Consumer/ManagementGroups	Creates a new Management Group. ManagementGroup object is required and it must have a Name, which must be unique. UsableId is ignored as it will be generated internally. Count and HashOfMembers are calculated internally and will be ignored if passed in the API call. Devices collection is optional and if provided should contain device FQDNs. Requires Tachyon Management Group support mode to be set to 'Internal'. **See note above**. **POST payload** `{ "ManagementGroup": { "Name": "Management Group A", "Description": "", "Expression": "" }, "Devices": ["machinename01.somedomain.com", "machinename02.somedomain.com"] }`	Requires 'Write' permission on on Management Groups
PUT	/Consumer/ManagementGroups	Updates an existing Management Group. Id must be a valid Id. You cannot change UsableId, Count or HashOfMembers Requires Tachyon Management Group support mode to be set to 'Internal'.	Requires 'Write' permission on on Management Groups

Verb	Request	Notes	Permissions required
		PUT Payload `{ "Id": 1, "Name": "Management Group A renamed", "Description": "", "Expression": "" }`	
DELETE	/Consumer/ManagementGroups/Id/{id}	Deletes a Management Group by its Id. Requires Tachyon Management Group support mode to be set to 'Internal'. New for version 8.0: An error will be returned if an attempt is made to delete a Management Group on which other entities have dependencies. This includes existence of other inherited Management Groups, Principal/Role assignments, assigned Policies, Surveys, Announcements and Event subscriptions.	Requires 'Write' permission on on Management Groups
DELETE	/Consumer/ManagementGroups/Name/{name}	Deletes a Management Group by its Name. Name needs to be supplied as Base64Encoded string. Requires Tachyon Management Group support mode to be set to 'Internal'. New for version 8.0: An error will be returned if an attempt is made to delete a Management Group on which other entities have dependencies. This includes existence of other inherited Management Groups, Principal/Role assignments, assigned Policies, Surveys, Announcements and Event subscriptions.	Requires 'Write' permission on on Management Groups
DELETE	/Consumer/ManagementGroups/UsableId/{usableId}	Deletes a Management Group by its UsableId.	Requires 'Write' permission on on

Verb	Request	Notes	Permissions required
		Requires Tachyon Management Group support mode to be set to 'Internal'. New for version 8.0: An error will be returned if an attempt is made to delete a Management Group on which other entities have dependencies. This includes existence of other inherited Management Groups, Principal/Role assignments, assigned Policies, Surveys, Announcements and Event subscriptions.	Management Groups
GET	/Consumer/ManagementGroups/Contents/Id/{id}	Returns all devices that belong to given Management Group using Management Group's Id.	Requires 'Read' permission on on Management Groups
GET	/Consumer/ManagementGroups/Contents/Name/{Name}	Returns all devices that belong to given Management Group using Management Group's Name. Name needs to be supplied as Base64Encoded string.	Requires 'Read' permission on on Management Groups
GET	/Consumer/ManagementGroups/Contents/UsableId/{usableId}	Returns all devices that belong to given Management Group using Management Group's UsableId.	Requires 'Read' permission on on Management Groups
POST	/Consumer/ManagementGroups/Contents	Adds device(s) to a Management Group. Payload for this request is an object representing a one-to-many relationship using Ids of objects.	Requires 'Write' permission on on Manage

Verb	Request	Notes	Permissions required

		Owner has the Id of the Management Group, Elements is an array of Ids of Devices.
		
		POST payload
		
		`{ "ManagementGroupId": 12, "DeviceFqdns": ["machinename01.somedomain.com", "machinename02.somedomain.com", "machinename03.somedomain.com", "machinename04.somedomain.com"] }`
		
		After devices have been added to the group Count and HashOfMembers for that group will be recalculated.
		
		Return payload will contain the Management Group in question and all devices that belong to it
		
		Sample Response payload Expand source
		
		`{ "ManagementGroup": { "Id": "12", "Name": "Management Group A", "Description": "", "Expression": "", "Count": "12", "UsableId": "0E9514B6-FD77-4077-9B54-A2AAA0C1953C", "HashOfMembers": "5c9f347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5b3ea" }, "Devices": [{ "Id":1, "Name":"machinename01", "Fqdn":"machinename01.somedomain.com", "Status":1, "OsType":"Windows",`
			ment Groups

Verb	Request	Notes	Permissions required
		"OsVerNum":2814750438195200, "OsVerTxt":"Microsoft Windows 10 Enterprise", "AgentVersion":2814835668418600, "Manufacturer":"Dell Inc.", "ChassisType":7, "DeviceType":"Desktop", "CpuType":"Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz", "CpuArchitecture":"x64", "OsArchitecture":"x64", "RamMB":32693, "SMBiosGuid":"4c4c4544-0051-5a10-8058-b7c04f47354a", "OsGuid":"4c4c4544-0051-5a10-8058-b7c04f47354a", "TachyonGuid":"4c4c4544-0051-5a10-8058-b7c04f47354a", "BiosVersion":"n/a", "LastBootUTC":1454408472, "LastConnUtc":1454604670, "CreatedUtc":"2016-02-03T17:20:55.003", "VrPlatform":"", "TimeZone":0, "CertType":null, "CertExpiryUtc":null, "PushToken":null, "Model":"Precision T5600", "Domain":"somedomain" }, { ... }, { ... }] } Requires Tachyon Management Group support mode to be set to 'Internal'.	
PUT	/Consumer/ManagementGroups/Contents	Replaces all the devices in a Management Group. All existing devices are deleted, and the	Requires 'Write' permissi

Verb	Request	Notes	Permissions required
		devices supplied with this request are stored in their place. After devices have been updated, the group Count and HashOfMembers for that group will be recalculated. The incoming payload and the return payload are exactly the same as specified above for a POST request.	on on Management Groups

DELETE	/Consumer/ManagementGroups/Contents	Removes device(s) from a Management Group.
		
		Payload for this request is an object representing a one-to-many relationship using Ids of objects.
		
		Owner has the Id of the Management Group, Elements is an array of Ids of Devices.
		
		DELETE payload
		
		{
		"ManagementGroupId": 12,
		"DeviceFqdns": ["machinename01.somedomain.com", "machinename02.somedomain.com"]
		}
		
		After devices have been removed from the group Count and HashOfMembers for that group will be recalculated.
		
		Return payload will contain the Management Group in question and all devices that belong to it.
		
		Sample Response payload Expand source
		
		{
		"ManagementGroup": {
		"Id": "1",
		"Name": "Management GroupA",
		"Description": "",
			Requires 'Write' permission on on Management Groups

Verb	Request	Notes	Permissions required


```
            "Expression": "",
            "Count": "12",
        "UsableId": "0E9514B6-FD77-407
7-9B54-A2AAA0C1953C",
            "HashOfMembers": "5c9f3474
57272d5edfeb5dd5c0f6fe4e38311b6126908f
2612a1e58d8fb5b3ea"
        },
        "Devices": [
                {
            "Id":1,
        "Name":"machinename03",
            "Fqdn":"machinename03.some
domain.com",
            "Status":1,
            "OsType":"Windows",
            "OsVerNum":281475043819520
0,
            "OsVerTxt":"Microsoft Wind
ows 10 Enterprise",
            "AgentVersion":28148356684
1860,
            "Manufacturer":"Dell Inc.
",
            "ChassisType":7,
            "DeviceType":"Desktop",
            "CpuType":"Intel(R) Xeon
(R) CPU E5-2630 0 @ 2.30GHz",
            "CpuArchitecture":"x64",
            "OsArchitecture":"x64",
            "RamMB":32693,
            "SMBiosGuid":"4c4c4544-005
1-5a10-8058-b7c04f47354a",
            "OsGuid":"4c4c4544-0051-5a
10-8058-b7c04f47354a",
            "TachyonGuid":"4c4c4544-00
51-5a10-8058-b7c04f47354a",
            "BiosVersion":"n/a",
            "LastBootUTC":1454408472,
            "LastConnUtc":1454604670,
            "CreatedUtc":"2016-02-03T1
7:20:55.003",
            "VrPlatform":"",
            "TimeZone":0,
            "CertType":null,
            "CertExpiryUtc":null,
```

Verb	Request	Notes	Permissions required

		<pre>"PushToken":null,
		"Model":"Precision T5600",
		"Domain":"somedomain"
		},
		{ ... },
		{ ... }
		]
		}</pre> Requires Tachyon Management Group support mode to be set to 'Internal'.	

POST	/Consumer/ManagementGroups/Clear/Id/{id}	Removes all Devices from a Management Group by the group's Id. Requires Tachyon Management Group support mode to be set to 'Internal'.	Requires 'Write' permission on on Management Groups
POST	/Consumer/ManagementGroups/Clear/Name/{name}	Removes all Devices from a Management Group by the group's Name. Name needs to be supplied as Base64Encoded string. Requires Tachyon Management Group support mode to be set to 'Internal'.	Requires 'Write' permission on on Management Groups
POST	/Consumer/ManagementGroups/Clear/UsableId/{usableId}	Removes all Devices from a Management Group by the group's UsableId. Requires Tachyon Management Group support mode to be set to 'Internal'.	Requires 'Write' permission on on Management Groups
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}	**Minimum API version 8.0**	Requires one of following:

Verb	Request	Notes	Permissions required
		Replaces /Consumer/ManagementGroups, allowing a filter by securable type and operation. By default this endpoint will return a flat list of Management Groups. An optional query string parameter called **view** can be used. If that parameter is set to **tree** this endpoint will return Management Groups as a tree structure. **NOTE:** If you make a request for "Security" securable type with either "Read" or "Write" operation API behaves differently. All Management Groups will be returned, not only those caller has Permission on, and every MG returned will have two extra fields are returned. A flag called "CallerHasPermissionToAccess", which indicates whether the caller has requested Permission on a given group and a property called "NumberOfAssignments", which has the number of RBAC assignments that use given Management Group. Using "Security" context is aimed at RBAC management, so creating assignments between Principals, Roles and Management Groups. The following is an example of what the tree looks like; for a plain list example, see /Consumer/ManagementGroups above: **Returned tree JSON** Expand source `[ { "Children": [ { "Children": [], "Id": 4,`	• Any permission on any of the InstructionSets • 'Read' perm

Verb	Request	Notes	Permissions required


```
                "Name": "UKDesktops",
                "Description": "All desktops in UK",
                "Expression": null,
                "Type": 1,
                "Count": 10,
                "UsableId": "DesktopsUK",
                "HashOfMembers": "AFCFAD58-49C5-45AE-A4F0-5C4EABA0F055",
                "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
                "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
                "ParentUsableId": "UK"
            },
            {
                "Children": [],
                "Id": 3,
                "Name": "UKServers",
                "Description": "All servers in UK",
                "Expression": null,
                "Type": 1,
                "Count": 10,
                "UsableId": "ServersUK",
                "HashOfMembers": "C74ACBD3-4BC2-4B97-9B79-568612D234F8",
                "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
                "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
                "ParentUsableId": "UK"
            }
        ],
        "Id": 2,
        "Name": "UK",
        "Description": "All computers in UK",
        "Expression": null,
        "Type": 1,
        "Count": 10,
        "UsableId": "UK",
        "HashOfMembers": "ADCA363B-C79F-4AA1-95EC-9357CE1FE88A",
```


isisonManagementGroups

Verb	Request	Notes	Permissions required


```
        "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
        "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
        "ParentUsableId": "global"
    },
    {
        "Children": [
            {
                "Children": [
                    {
                        "Children": [],
                        "Id": 9,
                        "Name": "USEastDesktops",
                        "Description": "All desktops in East US",
                        "Expression": null,
                        "Type": 1,
                        "Count": 10,
                        "UsableId": "DesktopsEastUS",
                        "HashOfMembers": "3B4D97DF-20B6-41F7-A333-7C1F400FD3F1",
                        "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
                        "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
                        "ParentUsableId": "EastUS"
                    },
                    {
                        "Children": [],
                        "Id": 8,
                        "Name": "USEastServers",
                        "Description": "All servers in East US",
                        "Expression": null,
                        "Type": 1,
                        "Count": 10,
```

Verb	Request	Notes	Permissions required


```
                                "UsableId": "S
erversEastUS",
                                "HashOfMembers
": "2697A98E-6808-4B39-BDDC-0C01C05514
83",
                                "CreatedTimest
ampUtc": "2021-05-18T16:20:11.47Z",
                                "ModifiedTimes
tampUtc": "2021-05-18T16:20:11.47Z",
                                "ParentUsableI
d": "EastUS"
                            }
                        ],
                        "Id": 6,
                        "Name": "USEast",
                        "Description": "All co
mputers in US East",
                        "Expression": null,
                        "Type": 1,
                        "Count": 10,
                        "UsableId": "EastUS",
                        "HashOfMembers": "7ECE
8794-7852-446F-B97D-EDE731CA4619",
                        "CreatedTimestampUtc":
"2021-05-18T16:20:11.47Z",
                        "ModifiedTimestampUtc
": "2021-05-18T16:20:11.47Z",
                        "ParentUsableId": "US"
                    },
                    {
                        "Children": [
                            {
                                "Children":
[],
                                "Id": 11,
                                "Name": "USWes
tDesktops",
                                "Description":
"All desktops in West US",
                                "Expression":
null,
                                "Type": 1,
                                "Count": 10,
                                "UsableId": "D
esktopsWestUS",
                                "HashOfMembers
```

Verb	Request	Notes	Permissions required


```
": "7E513BD0-DFDE-48A1-9A28-993D1E31CDE8",
                            "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
                            "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
                            "ParentUsableId": "WestUS"
                        },
                        {
                            "Children": [],
                            "Id": 10,
                            "Name": "USWestServers",
                            "Description": "All servers in West US",
                            "Expression": null,
                            "Type": 1,
                            "Count": 10,
                            "UsableId": "ServersWestUS",
                            "HashOfMembers": "1D5E80F5-8D2F-403A-8384-83AA8A5682EA",
                            "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
                            "ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
                            "ParentUsableId": "WestUS"
                        }
                    ],
                    "Id": 7,
                    "Name": "USWest",
                    "Description": "All computers in US West",
                    "Expression": null,
                    "Type": 1,
                    "Count": 10,
                    "UsableId": "WestUS",
                    "HashOfMembers": "9C2CAB68-2E91-4B1C-99AB-E5FEC8C26CE7",
                    "CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
```

Verb	Request	Notes	Permissions required

		<pre>"ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
		"ParentUsableId": "US"
		}
		],
		"Id": 5,
		"Name": "US",
		"Description": "All computers in US",
		"Expression": null,
		"Type": 1,
		"Count": 10,
		"UsableId": "US",
		"HashOfMembers": "29211754-F6E1-4286-BEA8-9BC1628C74BA",
		"CreatedTimestampUtc": "2021-05-18T16:20:11.47Z",
		"ModifiedTimestampUtc": "2021-05-18T16:20:11.47Z",
		"ParentUsableId": "global"
		}
		]</pre>	

GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Id/{id}	**Minimum API version 8.0** Replaces /Consumer/ManagementGroups/Id/{id} while providing additional filter options.	
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Name/{name}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Name/{name} while providing additional filter options.	
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/UsableId/{usableId}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/UsableId/{usableId} while providing additional filter options.	

Verb	Request	Notes	Permissions required
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Search/{base64SearchString}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Search/{base64SearchString} while providing additional filter options.	
POST	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Search	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Search while providing additional filter options.	
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Contents/Id/{id}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Contents/Id/{id} while providing additional filter options.	
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Contents/Name/{name}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Contents/Name/{name} while providing additional filter options.	
GET	/Consumer/ManagementGroups/SecurableType/{securableType}/Operation/{operation}/Contents/UsableId/{usableId}	**Minimum API version 8.0** Replaces Replaces /Consumer/ManagementGroups/Contents/UsableId/{usableId} while providing additional filter options.	
GET	/Consumer/ManagementGroups/AllDevices	**Minimum API version 8.0** Returns the 'All Devices' Management Group, which is identified in the database by the value "global" for the UsableId.	Requires one of following: • Any

Verb	Request	Notes	Permissions required
			permission on any of the InstructionSets • 'Read' permissio

Verb	Request	Notes	Permissions required
			nonManagementGroups
GET	/Consumer/ManagementGroups/AffectedEntities/UsableId/{usableId}	**Minimum API version 8.0** Returns a list of all entities that are linked to the Management Group. The use case is to present to the user a warning about the "links" to the MG that would prevent it from being deleted when the user requests deletion of the MG. **Sample output** `[ { "EntityName": "ManagementGroup", "EntityId": 4, "Name": "Servers in East US", "Reason": "Management Group 'Servers in East US' inherits from 'Computers in East US'" }, {`	

Verb	Request	Notes	Permissions required

		<pre>"EntityName": "PrincipalRoleManagementGroup",
		"EntityId": null,
		"Name": "Principal 'John Smith', Role 'Global Approvers'",
		"Reason": "There is an assignment for Principal 'John Smith', Role 'Global Approvers' to Management Group 'Computers in East US'"
		}
		]</pre>	

2.3.30 Notifications

Entity - Notifications - handles system notifications

Verb	Request	Notes	Permissions required
GET	/Consumer/Notifications	Returns all notifications **New for API version 8.0**: Only returns notifications where the *InstructionId* column is either NULL or refers to an instruction that is authorised for the invoking user in accordance with the rules for instruction visibility based on Management Groups (see the Instructions page).	'Read' permission to security
GET	/Consumer/Notifications/{instructionId}	Returns all notifications for a specific instruction **New for API version 8.0**: The list of notifications will come back empty if the selected instruction is not accessible to the user in accordance with the rules for instruction visibility based on	'Read' permission to security

Verb	Request	Notes	Permissions required
		Management Groups (see above).	
GET	/Consumer/Notifications/Component/{component}/{componentId:int?}	Returns all notifications for a specific component of the system or specific component type (so for instance for specific Switch or all Switches) **New for API version 8.0**: Only returns notifications where the *InstructionId* column is either NULL or refers to an instruction that is authorised for the invoking user in accordance with the rules for instruction visibility based on Management Groups (see the Instructions page).	'Read' permission to security
POST	/Consumer/Notifications	Returns notifications that fulfil conditions specified in the request body: • filter - filter expression.To learn about how to define filter checkUsing scope and filter expressionspage. • sort - sort expression. To learn about how to define sort criteria checkSort Definitionpage. • start - starting index to support pagination. It begins from 1.	'Read' permission to security

Verb	Request	Notes	Permissions required
		• pagesize - number of results to fetch Allowed filter columns: • Component • ComponentId • CreatedTime • Message • Operation • InstructionId • Type Allowed sort columns: • Component • ComponentId • Message • CreatedTime • Operation • InstructionId **New for API version 8.0**: Only returns notifications where the *InstructionId* column is either NULL or refers to an instruction that is authorised for the invoking user in accordance with the rules for instruction visibility based on Management Groups (see the Instructions page).	

2.3.31 Permissions

Entity - Role Based Access Control - Interrogates the Permission table in the database

Verb	Request	Notes	Permissions required

GET - single	/Consumer/Permissions/{permissionId }	Gets permission details for the specified Permission ID.
		
		Example Response
		
		{
		"Id": 15,
		"SecurableId": null,
		"SecurableTypeId": 3,
		"SecurableTypeName": "ProductPackManagement",
		"RoleId": 1,
		"RoleName": "Global Administrators",
		"OperationId": 9,
		"OperationName": "Add",
		"Allowed": true,
		"CreatedTimestampUtc": "2016-05-17T09:49:59.41Z",
		"ModifiedTimestampUtc": "2016-05-17T09:49:59.41Z"
		}
			• Read
			
			Returns Unauthorized (401) if user does not have the required permissions.

GET	/Consumer/Permissions/Role/{roleId }	Gets permission details for the specified Role ID. **Example Response** [{ "SecurableId": null, "SecurableName": null, "SecurableTypeId": 3, "SecurableTypeName": "ProductPackManagement", "RoleId": 2, "RoleName": "Product Pack Administrators", "Allowed": true, "Operations": [{ "PermissionId": 12, "OperationId": 9,	• Read Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required

		<pre>"OperationName": "Add",
		"CreatedTimestampUtc": "2016-05-17T09:49:59.41Z",
		"ModifiedTimestampUtc": "2016-05-17T09:49:59.41Z"
		},
		{
		"PermissionId": 13,
		"OperationId": 8,
		"OperationName": "Delete",
		"CreatedTimestampUtc": "2016-05-17T09:49:59.41Z",
		"ModifiedTimestampUtc": "2016-05-17T09:49:59.41Z"
		},
		{
		"PermissionId": 14,
		"OperationId": 10,
		"OperationName": "Read",
		"CreatedTimestampUtc": "2016-05-17T09:49:59.41Z",
		"ModifiedTimestampUtc": "2016-05-17T09:49:59.41Z"
		}
		]
		}
		]</pre>	

GET	/Consumer/Permissions/Principal/{accountName }	Gets permission details for the specified Account Name.
		
		AccountName should be specified in Base64 encoding.
		
		Example Response
		
		<pre>[
		{
		"SecurableId": 11,
		"SecurableName": null,
		"SecurableTypeId": 1,
		"SecurableTypeName": "ProductPack",
		"RoleId": 9,
		"RoleName": "CM questions",
		"Allowed": true,</pre>
			• Read
			
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required

		<pre>"Operations": [
		{
		"PermissionId": 24,
		"OperationId": 3,
		"OperationName": "Questioner",
		"CreatedTimestampUtc": "2016-05-18T15:56:46.09Z",
		"ModifiedTimestampUtc": "2016-05-18T15:56:46.09Z"
		}
		]
		}
		]</pre>
			A user can request own permissions.

GET	/Consumer/Permissions/Securable/{typeId}	Gets permission details for the specified securable type.	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/Permissions/Securable/{typeId}/{instanceId:int?}	Gets permission detailsfor the specified securable type instance. • TypeID = SecurableTypeId from the Permissions table • InstanceID = SecurableId as displayed in the Permissions table and also the Questions ID from the QuestionsDefinition table	• Read Returns Unauthorized (401) if user does not have the required

Verb	Request	Notes	Permissions required
			permissions.
GET	/Consumer/Permissions/Role/{roleId}/Type/{securableTypeName}/{instanceId:int?}	Gets permission details for the specified role on a securable type instance. • roleId - role id • securableTypeName - securable type • instanceId - securable instance id (optional)	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/Permissions/Principal/{accountName}/Type/{securableTypeName}/{instanceId:int?}	Gets permission details for the specified user account on a securable type instance. • accountName - user account • securableTypeName - securable type • instanceId - securable instance id (optional)	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/Permissions/RBAC/Enabled	Returns true if RBAC is enabled on the server, otherwise false.	• Unrestrict

Verb	Request	Notes	Permissions required
			ed
GET	/Consumer/Permissions/InstructionAccess/{instructionId}	Checks authorization of the calling user to access the specified instruction. Returns true if the calling user has Viewer permission on the Product Pack that contains the instruction definition for the specified instruction, otherwise returns false.	• Unrestricted
GET	/Consumer/Permissions/InstructionAccess/{InstructionId}/{managementGroupId}	**Minimum API version 8.0** Behaves like the previous API Consumer/Permissions/InstructionAccess/{instructionId} if the new (optional) parameter added at the end is omitted. But if provided, the instruction filters its result by the management group specified by this parameter.	• Unrestricted
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}	Checks authorization of the calling user to perform the specified operation. • securableType - type of object permissions is being tested • operation - operation on type/object for which permission is being tested	• Unrestricted
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}/{managementGroupId}	**Minimum API version 8.0** Behaves like the previous API /Consumer/Permissions/Type/{securableType}/Operation/{operation}	• Unrestrict

Verb	Request	Notes	Permissions required
		if the new (optional) parameter added at the end is omitted. But if provided, the instruction filters its result by the management group specified by this parameter.	
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}/{idProperty}/{id}	Checksauthorization of the calling user to access the specified securable type instance.. • securableType - type of object permissions is being tested • operation - operation on type/object for which permission is being tested • idProperty - identification property of type/object • id - identification value	• Unrestricted
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}/{idProperty}/{id}/{managementGroupId}	**Minimum API version 8.0** Behaves like the previous API /Consumer/Permissions/Type/{securableType}/Operation/{operation}/{idProperty}/{id} if the new (optional) parameter added at the end is omitted. But if provided, the instruction filters its result by the management group specified by this parameter.	• Unrestricted
POST	/Consumer/Permissions	Creates permissions in bulk.	• Write Returns Unauthorized (401) if user does not have the required

Verb	Request	Notes	Permissions required
			permissions.
POST	/Consumer/Permissions/single	Creates an individual permission. **Example Request** `{ "OperationName":"DummyOperation2", "SecurableTypeName":"MySecurableType", "RoleName":"Global Administrators", "Allowed":true }` • **OperationName** can be substituted by OperationId. These are mutually exclusive and only one can be used. • **SecurableTypeName** can be substituted by SecurableTypeId. These are mutually exclusive and only one can be used. • **RoleName** can be substituted by RoleId. These are mutually exclusive and only one can be used.	• Write Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/Permissions/{permissionId}	Deletes a permission by id.	• Write Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
GET	/Consumers/Permissions/ProductPack/{operation}	***REMOVED IN 3.1*** Checks if calling user has given permission on any of the product pack	None
GET	/Consumer/Permissions/InstructionSet/{operation}	**Minimum API version 3.1** Check if calling user has given permission on any instruction set	
GET	/Consumer/Permissions/InstructionSet/{operation}/{managementGroupId}	**Minimum API version 8.0** Behaves like the previous API /Consumer/Permissions/InstructionSet/{operation} if the new (optional) parameter added at the end is omitted. But if provided, the instruction filters its result by the management group specified by this parameter.	
GET	/Consumer/Permissions/{id}/ManagementGroups	**Minimum API version 3.3** Returns Management Groups linked to given permission **Obsolete as of API version 8.0**. Provided for backwards compatibility. Returns results by joining Permissions to Management Groups through the PrincipalRoleManagementGroup table, which may not match exactly the results returned by the previous version of the API.	Requires 'Read' on Security
POST	/Consumer/Permissions/{permissionId}/ManagementGroups/{managementGroupId}	**Minimum API version 3.3** Links given Management Group to given Permission. • permissionId - Id of the Permission • managementGroupId - Id of the Management Group	Requires 'Write' on Security

Verb	Request	Notes	Permissions required
		Return payload is empty. **Removed as of API version 8.0**. Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Permissions.	
POST	/Consumer/Permissions/ManagementGroups	**Minimum API version 3.3** Links given Management Groups to given Permission **Sample Request payload** `{ "PermissionId": 1, "ManagementGroupIds": [1,2,3] }` Return payload is empty. **Removed as of API version 8.0**. Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Permissions.	Requires 'Write' on Security
DELETE	/Consumer/Permissions/{permissionId}/ManagementGroups/{managementGroupId}	**Minimum API version 3.3** Removes a link between given Management Group from given Permission. • permissionId - Id of the Permission • managementGroupId - Id of the Management Group Return payload is empty. **Removed as of API version 8.0**. Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because	Requires 'Write' on Security

Verb	Request	Notes	Permissions required
		Management Groups are no longer linked to Permissions.	
DELETE	/Consumer/Permissions/ManagementGroups	**Minimum API version 3.3** Removes a link between given Management Groups from given Permission **Sample Request payload** `{ "PermissionId": 1, "ManagementGroupIds": [1,2,3] }` Return payload is empty. **Removed as of API version 8.0**. Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Permissions.	Requires 'Write' on Security
PUT	/Consumers/Permissions/refresh	Forces internal permissions cache to be refreshed.	None
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}/{idProperty}/{id}/UsableId/{managementGroupUsableId}	**Minimum API version 8.0** Similar to /Consumer/Permissions/Type/{securableType}/Operation/{operation}/{idProperty}/{id}/{managementGroupId} but accepting UsableId instead of regular Id	Checks authorization of the calling user to access the specified securable type instance
GET	/Consumer/Permissions/Type/{securableType}/Operation/{operation}/	**Minimum API version 8.0**	Checks authorization of

Verb	Request	Notes	Permissions required
	UsableId/{managementGroupUsableId}	Similar to /Consumer/Permissions/Type/{securableType}/Operation/{operation}/{managementGroupId:int?} but accepting UsableId instead of regular Id	the calling user to access the specified securable type instance

2.3.32 Policies

Entity - Policies

Minimum API version 4.0

Note for API version 8.0

As of API version 8.0, all the APIs that refer to the Policy Assignments "hide" from the caller any Management Groups on which the caller does not have any Role assignment. This is intended for delegation of administration where the user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results are the ones that you would get if the only existing Management Groups were those on which you have any Role assignment. In other words, trying to access a specific information about Policy assignments on which you have no rights on an assigned Management Group will return an error, and retrieving information about the Policy assignments will return only the assigned Management Groups that are accessible to the calling user.

Verb	Request	Notes	Permissions required
GET	/Consumer/Policies/{id}	Returns a single policy by its Id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policies/Pending/Changes	Returns information on	Requires 'Execute' permission on

Verb	Request	Notes	Permissions required
		whether policies have pending changes	'PolicyDeployment' securable type
GET	/Consumer/Policies	Returns all policies	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Policies/Search	Returns Policies that match the search parameters specified Allowed filter columns: • Name • Description • Enabled Allowed sort columns: • Name • Description • Enabled	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Policies/ShallowSearch	Returns Policies that match the search parameters specified. This endpoint returns Ids of rules instead of entire objects and Management	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		Group names instead of entire objects. Allowed filter columns: • Name • Description • Enabled Allowed sort columns: • Name • Description • Enabled	
POST	/Consumer/Policies	Creates a new policy	Requires 'Write' permission on 'GuaranteedState' securable type
POST	/Consumer/Policies/{id}/Rules	Adds given Rules to a specific policy and returns updated policy. Rule Ids should be passed in as an array of integers in the body of the request.	Requires 'Write' permission on 'GuaranteedState' securable type
DELETE	/Consumer/Policies/{id}/Rules	Removes all rules from a given	Requires 'Delete' permission on

Verb	Request	Notes	Permissions required
		policy and returns updated policy.	'GuaranteedState' securable type
DELETE	/Consumer/"Policies/{id}/Rules/{ruleId}	Removes a specific Rule from a given policy and returns updated policy.	Requires 'Delete' permission on 'GuaranteedState' securable type
PUT	/Consumer/Policies	Updates and existing policy and returns updated policy.	Requires 'Write' permission on 'GuaranteedState' securable type
DELETE	/Consumer/Policies/{id}	Deletes a policy	Requires 'Delete' permission on 'GuaranteedState' securable type
POST	/Consumer/Policies/{id}/Enable/{enable}	Enables a policy if 'enable' parameter is set to 'true' or disables a policy if 'enable' parameter is set to 'false'.	Requires 'Write' permission on 'GuaranteedState' securable type
POST	/Consumer/Policies/{id}/Assignments	Assigns a Policy to selected Management groups.	Requires 'Write' permission on 'GuaranteedState' securable type
		Management Groups Ids should be passed in as an array of integers in the body of the request.	New for API version 8.0: Requires 'Assign' permission on the 'PolicyAssignment' securable type

Verb	Request	Notes	Permissions required
PUT	/Consumer/Policies/{id}/Assignments	Replaces all Management Groups assigned to a given policy with Management Groups supplied in this request. Management Groups Ids should be passed in as an array of integers in the body of the request.	Requires 'Write' permission on 'GuaranteedState' securable type New for API version 8.0: Requires 'Assign' permission on the 'PolicyAssignment' securable type
GET	/Consumer/Policies/{id}/Assignments	Returns Management groups assigned to given Policy	Requires 'Read' permission on 'GuaranteedState' securable type
DELETE	/Consumer/Policies/{id}/Assignments	Removes all Management Groups from given policy	Requires 'Delete' permission on 'GuaranteedState' securable type New for API version 8.0: Requires 'Assign' permission on the 'PolicyAssignment' securable type
DELETE	/Consumer/Policies/{id}/Assignments/ManagementGroups/{managementGroupId}	Removes specific Management Group from given policy	Requires 'Delete' permission on 'GuaranteedState' securable type New for API version 8.0: Requires 'Assign' permission on the

Verb	Request	Notes	Permissions required
			'PolicyAssignment' securable type
GET	/Consumer/Policies/Unlicensed	Minimum API version 5.1 Returns a list of Ids of Policies that use unlicensed Fragments	Requires 'Read' permission on 'GuaranteedState' securable type

2.3.33Policy

Entity - Policy

Minimum API version 4.0

Verb	Request	Notes	Permissions required
GET	/Consumer/Policy/Changes	Minimum API version 5.1 Returns the list of changes that will be deployed in case a policy Deployment is requested at this time.	Requires 'Execute' permission on 'PolicyDeployment' securable type
POST	/Consumer/Policy	Initiates Policy deployment.	Requires 'Execute' permission on 'PolicyDeployment' securable type
POST	/Consumer/Policy/Responses/{ruleId}	Returns responses to a Policy for a specific Rule.	Requires 'Read' permission on

Verb	Request	Notes	Permissions required
		Results can be filtered and sorted, they can also be retrieved in batches. **Minimum information required in the request payload** `{ "PageSize":20, "Start":"0;0" }` Apart from the minimum required you can see in the example above, the request can contain other fields: • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId.	'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		• **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page. • **ClobDefaultReadSize** - Number of characters to which a CLOB type field will be truncated. If this value is not provided a global value configured in GlobalSettings will be used. Please see Instruction Definition Reference#SchemaJson page for details	
POST	/Consumer/Policy/Responses/Search	Returns responses to policies. Results can be filtered and sorted, they can also be retrieved in batches. Request payload has the same fields as in the request above.	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Policy/Responses/{policyId:int}/Search	Return responses to a specific policy. Results can be filtered and sorted, they can also be retrieved in batches.	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/DeviceStatus	Returns dashboard device status	Requires 'Read' permission on 'Guarantee

Verb	Request	Notes	Permissions required
			dState' securable type
GET	/Consumer/Policy/Dashboard/DeviceStatus/{id}	Returns dashboard device status for a given policy	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/DeviceStatusByCriticality	Returns dashboard device status by criticality	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/DeviceStatusByCriticality/{id}	Returns dashboard device status by criticality for a given policy id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/PolicyRuleStatus	Returns dashboard policy rule status	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
GET	/Consumer/Policy/Dashboard/PolicyRuleStatus/{id}	Returns dashboard policy rule status for a given policy id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/PolicyEffectiveness	Returns dashboard with policy effectiveness	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/PolicyEffectiveness/{id}	Returns dashboard with policy effectiveness for a given policy id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/PolicyRuleRemediation/{days}	Returns dashboard device status in the last X days	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Policy/Dashboard/PolicyRuleRemediation/{days}/{id}	Returns dashboard device status for a given policy id	Requires 'Read' permission on

Verb	Request	Notes	Permissions required
			'GuaranteedState' securable type
POST	/Consumer/Policy/View/PolicyStatus/Search	Returns filtered and sorted policy statuses. • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
POST	/Consumer/Policy/View/DevicePolicyRuleStatus/Search	Returns filtered and sorted policy rule status • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Policy/View/DevicePolicyRuleStatus/ policyid/{policyId}/Search	Returns filtered and sorted policy rule status • **Filter** - filter expression. If null then no filter is applied. Searchable columns	Requires 'Read' permission on 'GuaranteedState'

Verb	Request	Notes	Permissions required
		depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	securable type
POST	/Consumer/Policy/View/PolicyRuleStatus/Search	Returns filtered and sorted rules status • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page.	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		• **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	
POST	/Consumer/Policy/View/PolicyRuleStatus/ policyid/{policyId}/Search	Returns filtered and sorted rules status • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	
POST	/Consumer/Policy/View/PolicyRuleStatus/ tachyonguid/{guid}/Search	Returns filtered and sorted rules for given device • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	
POST	/Consumer/Policy/View/PolicyRuleStatus/policyid/{policyId}/tachyonguid/{guid}/Search	Returns filtered and sorted rules for given device • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	
POST	/Consumer/Policy/View/DevicePolicyRuleStatus/ ruleid/{ruleId}/Search	Returns filtered and sorted device rules • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
POST	/Consumer/Policy/View/DevicePolicyRuleStatus/policyid/{policyId}/ruleid/{ruleId}/Search	Returns filtered and sorted device rules • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Policy/View/DevicePolicyPolicyStatus/policyid/{policyId}/Search	Returns filtered and sorted policy device from policy status • **Filter** - filter expression. If null then no filter is applied. Searchable columns	Requires 'Read' permission on 'GuaranteedState'

Verb	Request	Notes	Permissions required
		depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	securable type
POST	/Consumer/Policy/View/PolicyRulePolicyDeviceStatus/policyid/{policyId}/tachyonguid/{guid}/Search	Returns filtered and sorted policy rule device and policy status • **Filter** - filter expression. If null then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page.	Requires 'Read' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions required
		• **Pagesize** - number of results to fetch. • **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded Range value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list of pairs of ShardId;ResponseId. • **Sort** - Defines sorting parameters. Data is sorted after it is filtered but before it is paginated. More on sorting can be found on Sort Definition page.	

2.3.34 PolicyAssignments

Entity - Policy Assignments

Minimum API version 4.0

Note for API version v8.0

As of API version 8.0, all the APIs that refer to the Policy Assignments "hide" from the caller any Management Groups on which the caller does not have any Role assignment. This is intended for delegation of administration where the user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but the returned results are the ones that you would get if the only existing Management Groups were those on which you have any Role assignment. In other words, trying to access a specific information about Policy assignments on which

you have no rights on an assigned Management Group will return an error, and retrieving information about the Policy assignments will return only the assigned Management Groups that are accessible to the calling user.

Verb	Request	Notes	Permissions required
POST	/Consumer/PolicyAssignments	Assigns a Policy to a Management Group **Payload** `{ "PolicyId": 1, "ManagementGroupId": 12 }`	Requires 'Write' permission on 'GuaranteedState' securable type New for API version 8.0: Requires 'Assign' permission on the 'PolicyAssignment' securable type
GET	/Consumer/PolicyAssignments/{id}	Returns a specific Policy assignment object by its Id	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/PolicyAssignments	Returns all policy assignments	Requires 'Read' permission on 'GuaranteedState' securable type
DELETE	/Consumer/PolicyAssignments/{id}	Deletes a specific Policy assignment object by its Id	Requires 'Delete' permission on 'GuaranteedState' securable type New for API version 8.0: Requires 'Assign' permission on the 'PolicyAssignment' securable type

2.3.35PrincipalRoleManagementGroups

Entity - Role Based Access Control - Interrogates the PrincipalRoleManagementGroup table in the database

All the entries described below require at least **API version 8.0**.

Verb	Request	Notes	Permissions Required

GET	/Consumer/PrincipalRoleManagementGroups	Gets all assignments that are visible to the invoking user.
		
		Example data returned
		Expand source
		
		[
		{
		"PrincipalId": 1,
		"RoleId": 1,
		"ManagementGroupId": 1,
		"CreatedTimestampUtc": "2021-04-15T11:25:49.423Z",
		"Principal": {
		"Id": 1,
		"ExternalId": "S-1-5-21-193489370-1251057138-4208286054-1234",
		"PrincipalName": "Domain\\User",
		"Email": null,
		"Enabled": true,
		"CreatedTimestampUtc": "2020-02-21T09:23:31.937Z",
		"ModifiedTimestampUtc": "2020-02-21T09:23:31.937Z",
		"SystemPrincipal": false,
		"DisplayName": "Alberto",
		"IsGroup": false
		},
		"Role": {
		"AssignedManagementGroupCount": 2,
		"HasAllDevi
			Requires 'Read' permission on the 'Security' securable type

Verb	Request	Notes	Permissions Required


```
cesManagementGroupAssig
ned": true,
            "AssignedPr
incipalCount": 2,
            "Id": 1,
            "Name": "Gl
obal Administrators",
            "Descriptio
n": "Has the combined r
ights of all the other
system roles",
            "CreatedTim
estampUtc": "2020-02-21
T09:23:31.907Z",
            "ModifiedTi
mestampUtc": "2020-04-1
4T15:30:02.96Z",
            "SystemRole
": true
        },
        "ManagementGrou
p": {
            "Id": 1,
            "Name": "Al
l Devices",
            "Descriptio
n": "All devices are me
mbers of this Managemen
tGroup",
            "Expression
": null,
            "TachyonMan
agementGroupType": 0,
            "TachyonDev
iceCount": -1,
            "UsableId":
"global",
            "HashOfMemb
ers": "global",
            "CreatedTim
estampUtc": "2020-02-21
T09:23:34.23Z",
            "ModifiedTi
mestampUtc": "2020-02-2
1T09:23:34.23Z",
            "ParentUsab
```

Verb	Request	Notes	Permissions Required

		<pre>leId": null
		        }
		    },
		    {
		        ...etc...
		    }
		]</pre>	

GET	/Consumer/PrincipalRoleManagementGroups/ Principal/Id/{principalId}	Gets all assignments for a Principal given by its Id	Requires 'Read' permission on the 'Security' securable type
GET	/Consumer/PrincipalRoleManagementGroups/ Principal/Name/{principalName}	Gets all assignments for a Principal given by its Name	Requires 'Read' permission on the 'Security' securable type
GET	/Consumer/PrincipalRoleManagementGroups/ Role/Id/{roleId}	Gets all assignments for a Role given by its Id	Requires 'Read' permission on the 'Security' securable type
GET	/Consumer/PrincipalRoleManagementGroups/ Role/Name/{roleName}	Gets all assignments for a Role given by its Name	Requires 'Read' permission on the 'Security' securable type

Verb	Request	Notes	Permissions Required
GET	/Consumer/PrincipalRoleManagementGroups/ManagementGroup/Id/{managementGroupId}/ {includeInherited}	Gets all assignments for a Management Group given by its Id The optional parameter *includeInherited* is a true/false value (defaults to *false* if not provided in the query string) that determines whether the results will include assignments that are not explicitly provided for the given Management Group, but rather are inferred from the Management Groups inheritance tree. A property called IsInherited is returned in every row to indicate if the row is inferred from inheritance or is an actual explicit assignment.	Requires 'Read' permission on the 'Security' securable type
GET	/Consumer/PrincipalRoleManagementGroups/ManagementGroup/UsableId/{usableId}/ {includeInherited}	Gets all assignments for a Management Group given by its UsableId The optional parameter *includeInherited* is a true/false value (defaults to *false* if not provided in the query string) that determines whether the results will include assignments that are not explicitly provided for the given Management Group, but rather are inferred from the Management Groups inheritance tree. A property	Requires 'Read' permission on the 'Security' securable type

Verb	Request	Notes	Permissions Required
		called IsInherited is returned in every row to indicate if the row is inferred from inheritance or is an actual explicit assignment.	
POST	/Consumer/PrincipalRoleManagementGroups	Adds multiple assignments, ignoring any already existing entries. Returns a collection containing the newly created assignments. **Sample input** `[ { "PrincipalId":3, "RoleId": 10, "ManagementGroupId": 5 }, { "PrincipalId":3, "RoleId": 10, "ManagementGroupId": 7 } ]`	Requires 'Write' permission on the 'Security' securable type
PUT	/Consumer/PrincipalRoleManagementGroups/ Principal/Id/{principalId}	Replaces all current assignments for a Principal with the set supplied in the request. All entries in the set are assigned to the principal indicated in the query string. Any principal Id specified within the entries is ignored. **Example input body** `[ { "RoleId": 10, "ManagementGroupId": 5 }, { "RoleId": 10, "ManagementGroupId": 7 ]`	Requires 'Write' permission on the 'Security' securable type

Verb	Request	Notes	Permissions Required
		Returns the newly created assignments.	
PUT	/Consumer/PrincipalRoleManagementGroups/ Principal/Name/{principalName}	Like above, but the Principal that receives the assignments is specified by Name instead of by Id	Requires 'Write' permission on the 'Security' securable type
PUT	/Consumer/PrincipalRoleManagementGroups/ Role/Id/{roleId}	Replaces all current assignments for a Role with the set supplied in the request. All entries in the set must be for the same role.	Requires 'Write' permission on the 'Security' securable type
PUT	/Consumer/PrincipalRoleManagementGroups/ Role/Name/{roleName}	Like above, but the Role that receives the assignments is specified by Name instead of by Id	Requires 'Write' permission on the 'Security' securable type
PUT	/Consumer/PrincipalRoleManagementGroups/ ManagementGroup/Id/{Id}	Replaces all current assignments for a Management Group with the set supplied in the request. All entries in the set must be for the same MG.	Requires 'Write' permission on the 'Security' securable type
PUT	/Consumer/PrincipalRoleManagementGroups/ ManagementGroup/UsableId/{usableId}	Like above, but the Management Group that receives the assignments is	Requires 'Write' permission on the 'Security'

Verb	Request	Notes	Permissions Required
		specified by UsableId instead of by Id	securable type
DELETE	/Consumer/PrincipalRoleManagementGroups	Deletes multiple assignments, specified in the body of the request. Input: takes an array of PrincipalRoleManagementGroup objects.	Requires 'Write' permission on the 'Security' securable type
DELETE	/Consumer/PrincipalRoleManagementGroups/PrincipalId/{principalId}/RoleId/{roleId}/ManagementGroupId/{managementGroupId}	Deletes a single assignment, specified in the URL.	Requires 'Write' permission on the 'Security' securable type

2.3.36 PrincipalRoles

Entity - Role Based Access Control - Interrogates the PrincipalRole table in the database

Verb	Request	Notes	Permissions Required

POST	/Consumer/PrincipalRoles	Creates a new link for Roles to Principals
		
		Example Request
		
		{
		"PrincipalId": 1,
		"RoleId": 2
		}
		
		Obsolete as of API version 8.0. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume
			• Write
			
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions Required
		Management Group was set to "All Devices".	

POST	/Consumer/PrincipalRoles/SavePrincipal	Creates links between a principal and multiple roles.
		
		Example Request
		
		{
		"PrincipalId": 1,
		"RoleIdList": [5, 8, 10]
		}
		
		Obsolete as of API version 4.0
		
		Version 8.0: Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".
			• Write
			
			Returns Unauthorized (401) if user does not have the required permissions.

DELETE	/Consumer/PrincipalRoles/Role/{RoleId}/Principal/{PrincipalId }	Deletes a link from the PrincipalRoles table when there is a match on Role ID and Principal ID. **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	• Delete Returns Unauthorized (401) if user does not have the required permissions.
POST	/Consumer/PrincipalRoles/Principal/{principalId}	Creates links between a principal and multiple roles. principalId (URI) - the Id of the principal that should be assigned the roles.	Requires 'Write' on 'Security' securable type

Verb	Request	Notes	Permissions Required
		roleIds (BODY) - an array of Ids for roles that should be assigned to the principal **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	
PUT	/Consumer/PrincipalRoles/Principal/{principalId}	Replaces existing links between a principal and all of that principal's roles with links to roles passed in the body of the request. principalId (URI) - the Id of the principal roleIds (BODY) - an array of Ids for roles that should be assigned to the principal **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	Requires 'Write' on 'Security' securable type
DELETE	/Consumer/PrincipalRoles/Principal/{principalId}	Removes links between a principal and multiple roles. principalId (URI) - the Id of the principal that should be unassigned from the roles.	Requires 'Write' on 'Security' securable type

Verb	Request	Notes	Permissions Required
		roleIds (BODY) - an array of Ids for roles that should be unassigned from the principal **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	
POST	/Consumer/PrincipalRoles/Role/{roleId}	Creates links between a role and multiple principals. roleId (URI) - the Id of the role that the principals should be assigned to principal principalIds (BODY) - an array of Ids for principals that should be assigned to the role **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	Requires 'Write' on 'Security' securable type
PUT	/Consumer/PrincipalRoles/Role/{roleId}	Replaces existing links between a role and all of that role's principals with links to roles passed in the body of the request. roleId (URI) - the Id of the role	Requires 'Write' on 'Security' securable type

Verb	Request	Notes	Permissions Required
		principalIds (BODY) - an array of Ids for principals that should be assigned to the role **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	
DELETE	/Consumer/PrincipalRoles/Role/{roleId}	Removes links between a role and multiple principals. roleId (URI) - the Id of the role that should be unassigned from the principals. principalIds (BODY) - an array of Ids for principals that should be unassigned from the role **Minimum API version 4.0** **Obsolete as of API Version 8.0**. Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent and will assume Management Group was set to "All Devices".	Requires 'Write' on 'Security' securable type

2.3.37Principals

Entity - Role Based Access Control - Interrogates the Principal table in the database

Verb	Request	Notes	Permissions required
GET - all	/Consumer/Principals	Gets list of all Principals (users) and their details	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET - single	/Consumer/Principals /{principalId}	Returns all the details for the specified Principal ID	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET - for specific role	/Consumer/Principals /Role/{roleId}	Returns all principals for a specific role	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/Principals/ PermissionsAdmins	Returns all principals who have write permission to security	None
POST	/Consumer/Principals	Creates a Principal record. External Id is the Active Directory SID for the account or group. **Example Request** `{ "ExternalId": "8E527555-2D1C-4CD5-AC55-1B44F314DC65", "PrincipalName": "dev101\TchynAdmin",`	• Write Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required

		<pre> "Email": "TchynAdmin@dev101.local"
		}</pre>
		Example Response
		<pre>{
		  "Id": 17,
		  "ExternalId": "8E527555-2D1C-4CD5-AC55-1B44F314DC65",
		  "PrincipalName": "dev101\TchynAdmin",
		  "Email": "TchynAdmin@dev101.local",
		  "CreatedTimestampUtc": "2015-09-16T14:04:05.8636311Z",
		  "ModifiedTimestampUtc": "2015-09-16T14:04:05.8636311Z"
		}</pre>	

PUT	/Consumer/Principals	Modifies a Principal record.
		
		External Id is the Active Directory SID for the account or group.
		
		Example Request
		<pre>{
		  "Id": 17,
		  "ExternalId": "8E527555-2D1C-4CD5-AC55-1B44F314DC65",
		  "PrincipalName": "dev101\TchynAdmin",
		  "Email": "TchynAdmin@dev101.local"
		}</pre>
		Example Response
		<pre>{
		  "Id": 17,
		  "ExternalId": "8E527555-</pre>
			• Write
			
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		2D1C-4CD5-AC55-1B44F314DC65", "PrincipalName": "dev101\TchynAdmin", "Email": "TchynAdmin@dev101.local", "CreatedTimestampUtc": "2015-09-16T14:04:05.86", "ModifiedTimestampUtc": "2015-09-16T14:04:06.8776311Z" }	
DELETE	/Consumer/Principals/{principalId}	Delete the Principal ID if it exists. Otherwise "ExceptionMessage": "Principal record with Id={id} was not found" is returned. New in version 8.0: An error will be returned if deleting a Principal that has any assignments to Management Groups / Roles.	• Delete Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/Principals	**Minimum API version 4.0** Deletes multiple principals. System principals cannot be deleted. Body of the request should contain a collection of principals Ids in an array. New in version 8.0: An error will be returned if deleting a Principal that has any assignments to Management Groups / Roles.	Requires 'Delete' permission on 'Security' securable type

Verb	Request	Notes	Permissions required
PUT	/Consumer/Principals/Enable	**Minimum API version 4.0** Enables multiple principals. Body of the request should contain a collection of principals Ids in an array.	Requires 'Write' permission on 'Security' securable type
PUT	/Consumer/Principals/Disable	**Minimum API version 4.0** Disables multiple principals. Body of the request should contain a collection of principals Ids in an array.	Requires 'Write' permission on 'Security' securable type

2.3.38 PrincipalSearch

Entity - Role Based Access Control - Interrogates the User and Groups in Active Directory

Verb	Request	Notes	Permissions required

GET	/Consumer/PrincipalSearch/ {encoded search string}	Search string must be Base64 encoded
		
		Returns a UserSearchItem class:
		
		Return class
		
		`public class UserSearchItem`
		`{`
		`  UserName`
		`  ExternalId`
		`  Email`
		`  FullName`
		`}`
		
		Note: Only returns principals that are NOT already present in the Tachyon Principal table. The reason is that this API is intended to be used by the UI that adds Principals to Tachyon.
			• Read
			
			Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		Note: Obsolete from 5.0 onwards. Use POST to /Consumer/PrincipalSearch instead	
POST	/Consumer/PrincipalSearch	**Minimum API version 5.02** Searches for an active directory object matching the SearchText. **Request payload** `{ "SearchText": "user", "ObjectTypes": [ "user", "group" ], "PageSize": 25, "Sort": { "Column": "displayname", "Direction": "ASC" } }` Payload fields: • SearchText - text to search for. Mandatory field. Must not be empty. • ObjectTypes - types of AD/LDAP objects to search for. Allowed values are 'user' and 'group'. Mandatory field. Must contain at least one valid value. • PageSize - number of results to return from AD/LDAP. This limit is applied to the AD/LDAP search itself and you might be receiving even fewer results if any of those returned by the AD/LDAP search already exist within Tachyon. Optional parameter. Default value is 100. • Sort - sorting specification. Only sorting on a single column is supported. Allowed columns: cn, mail, sAMAccountName, description, objectSid, displayName. Allowed directions:	• Read Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		"ASC", "DESC". If not provided default sorting is ascending on displayName. **Response** `[ { "PrincipalName": "1E\\User.Name", "ExternalId": "S-1-5-21-9999999999-888888888-7777777777-66666", "Email": "user.name@1e.com", "DisplayName": "User Name", "IsGroup": false } ]`	
GET	/Consumer/PrincipalSearch/User/ {encoded search string}	Search string is a user name such as "DOMAIN\USER", and it must be sent encoded in Base64 Returns a UserSearchItem class: **Response** `[ { "PrincipalName": "1E\\User.Name", "ExternalId": "S-1-5-21-9999999999-888888888-7777777777-66666", "Email": "user.name@1e.com", "DisplayName": "User Name", "IsGroup": false } ]` Note: At this time, the API always returns a single user, but the result is an array in case future versions become able to return multiple users. An empty array will be returned if the user is not found. The API attempts to find the user in the Tachyon Principal table. Failing that, the groups of the user are listed from AD and matched against the tachyon Principal table. If the user belongs to a group that is a	• Read Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		valid Tachyon Principal, then the user's information is returned.	
GET	/Consumer/PrincipalSearch/ GetMembers/{groupName}	Returns members (users and groups, but not computers) of a specified Active Directory group. Name of the group must be base64 encoded	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/PrincipalSearch/ DisplayName/{accountName}	Returns display name.	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/PrincipalSearch/ WhoAmI	**Minimum API version 3.1** Returns currently logged in user info. **Response Payload**	N/A

Verb	Request	Notes	Permissions required

		<pre>{
		  "PrincipalName": "domain\\user.name",
		  "ExternalId": "S-1-5-21-12028960629-789336058-1343024091-20287",
		  "Email": "user.name@domain.com",
		  "DisplayName": "User Name",
		  "Photo": "base64photo"
		}</pre>	

GET	/Consumer/PrincipalSearch/Whois/{accountname}	**Minimum API version 5.1** Looks up a user in the Active Directory using the account name provided, and returns extended information about the user. Account name must be Base64 encoded. Use Url-Safe Base64 ('_' instead of '/', '-' instead of '+', remove trailing '='). The account name format is expected to be DOMAIN\USER.	• Read

2.3.39 ProductPacks

Entity - Product Packs

Verb	Request	Notes	Permissions required
POST	/Consumer/ProductPacks	Uploads a product pack or xml instruction file without assigning any uploaded instructions to an instruction set (which makes them unusable until an administrator assigns them).	Requires 'Add' permission on instructionSet management
POST	/Consumer/ProductPacks/InstructionSet/Id/{id}	**Minimum API version - 3.1**	Requires 'Add' permission on

Verb	Request	Notes	Permissions required
		Uploads a product pack or xml instruction file and assigns all uploaded instructions to a specific instruction set (specified by its {id}).	instruction Set management
GET	/Consumer/ProductPacks /{ProductPackId}	***REMOVED IN API VERSION 3.1*** Get a single Product Pack **Example Response** `{ "Id": 27, "Name": "Security Pack", "Description": "For securing all the things, securely.", "Icon": "iVBORw0KGgoAAA.....ErkJggg==", "Version": 2, "UploadTimestampUtc": "2015-08-14T09:29:43.707" }`	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/ProductPacks	***REMOVED IN API VERSION 3.1*** Get all the available Product Packs	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/ProductPacks /Name/{name}	***REMOVED IN API VERSION 3.1*** Get a single product pack by its name.	• Read Returns Unauthorized (401) if user does not have

Verb	Request	Notes	Permissions required
			the required permissions. • Add
POST	/Consumer/ProductPacks	**DOCUMENTATION FOR VERSION 3.1 AND ONWARDS** Uploads a single file, either an xml file with an instruction definition or a zip file containing multiple instruction definitions each in a separate xml file. Upload is done in a single transaction and entire process fails if any of the stages fail: • Instruction definitions are verified and resources extracted from instruction definitions • New entries in Instruction Definition table are created • Task groups and Tasks are created • Resources are sent to background API(s) **DOCUMENTATION FOR VERSIONS PRIOR TO 3.1** Upload a single Product Pack zip file,which includes an XML manifest to define and validate the payload. Optional query string parameter - force, can be used to force update of an existing product pack that has instructions in progress. If this flag is absent the upload will fail if the product pack that is supposed to be updated has instructions in progress. To force an update use: /Consumer/ProductPacks?force=true	Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		The following actions are performed, with termination of the upload if any stages fail: • The Product Pack file resources are posted to the URL of each registered Background Channel. • A new **Product Pack** table entry is created, this includes an Product Pack icon for UI use. • New questions and actions are added to the **Question Definition** table, referencing the new Product Pack Id. • New file resources are added as **Product Pack Resource** table entries, again referencing the new Product Pack Id. Note: When uploading the Product Pack via Postman ensure there are no Headers and Body just contains form-data and File is uploaded with Key=Filename	
DELETE	/Consumer/ProductPacks /{ProductPackId}	***REMOVED IN API VERSION 3.1*** Remove a single Product Pack. This includes the following actions: • The Product Pack file resources are deleted from each registered Background Channel, by DELETE sent to each URL. • The file resources associated with this Product Pack Id are deleted from the **Product Pack Resource** table. • Questions and actions associated with this Product Pack Id are deleted from the **Question Definition** table.	• Delete Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required
		• The **Product Pack** table entry for this Product Pack Id is deleted. If there are instructions in progress based on definitions that came from the pack you want to delete, an error will be returned. You can force the deletion and cancel those instructions by using the optional force query string parameter like so: **/Consumer/ProductPacks /{ProductPackId}?force=true**	
GET	/Consumer/ProductPacks/ {ProductPackId}/Content	***REMOVED IN API VERSION 3.1*** Download saved product pack's contents as ZIP file. If no content found, NO CONTENT (204) HTTP response status is returned.	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/ProductPacks/ {id}/Icon	***REMOVED IN API VERSION 3.1*** Gets a Product Pack's icon in binary format.	None

2.3.40 PushMessageUsers (removed)

Entity - PushMessageUsers - manages users registered for push message notifications

API versions up to and including 3.3. Unavailable in versions 4.0 onwards.

Verb	Request	Notes	Permissions required
GET	/Consumer/PushMessageUsers	Returns all user registered for push notifications **Return JSON**	"Read" permissions to Security

Verb	Request	Notes	Permissions required

		<pre>{
		"Id": 1,
		"DeviceType" : "Android",
		"UserName" : "somecompany\someuser",
		"CreatedTimeStampUtc": "2016-12-21 09:56:32.837",
		"ModifiedTimestampUtc": "2016-12-21 09:56:32.837",
		"Status": 1
		}</pre>	

GET	/Consumer/PushMessageUsers/{id}	Returns a specific user (if one exists) registered for push message notifications. Return data as above	"Read" permissions to Security
PUT	/Consumer/PushMessageUsers	Updates specified user. In/out payload as above At the moment you can only change the Status field and the structure you pass in must have a valid user Id. All other fields remain the same and Modified date is set to time when the modification took place. As such, any data passed in those fields will be ignored	"Write" permissions to Security

2.3.40.1 User status values:

1 - Active

2 - Disabled

3 - Deleted

2.3.41 RelatedInstructionDefinitions

Entity - Related Instruction Definitions

Verb	Request	Notes	Permissions required
GET	/Consumer/RelatedInstructionDefinitions/Id/{id}?instructionType={type}	Gets instruction definitions of specified type(s) related to an instruction definition. For example: http://<myserver>//Consumer/RelatedInstructionDefinitions/Id/2?instructionType=Action This example returns a list of instruction definitions of Action type related to instruction definition with id 2. More than one type can be mentioned in the query string. **If no type is mentioned, all related instruction definitions are returned.**	Questions • Questioner • Actioner Actions • Actioner It does not return Unauthorized (401) rather filters out instruction definitions based on permissions

2.3.42 Resources

Entity - Resources - Returns instruction resources stored in the database

Note

These APIs are NOT YET RELEASED. We expect them to be present in either version 8.1 or maybe 8.2. No changes are expected, but since the work on these features is still in progress there is a small chance that the final APIs could be slightly different from what is currently documented here. The page will be updated if this happens.

Verb	Request	Notes	Permissions required
GET	/Consumer/Resources/Content/{fileName}	Gets the binary content of a Resource based on the file name field that exists in the database. The fileName has to be base64 encoded using Url-Safe Base64. This API only exists for compatibility with legacy Agents. Use the variant below for new code.	Currently un-permissioned. This will change before this API is RTM'ed.
GET	/Consumer/Resources/Content/{filename}/{hash}/{size}/{platform}	Gets the binary content of a Resource based on the set of fields that uniquely identify the Resource in the database. The fileName has to be base64 encoded using Url-Safe Base64. The rest of the parameters should match the corresponding field in the database. The hash is a sequence of 64 hex digits without punctuation.	Currently un-permissioned. This will change before this API is RTM'ed.
GET	/Consumer/Resources/PolicyDocuments/{hash}	Gets the binary content of a PolicyDocument based on its hash. The hash should be passed as a string of 64 hex digits.	Currently un-permissioned. This will change before this API is RTM'ed.

Verb	Request	Notes	Permissions required
GET	/Consumer/Resources/Installers/{id}	Gets the binary content of an Agent Installer based on the id with which it is stored in the database.	Currently un-permissioned. This will change before this API is RTM'ed.

2.3.43 ResponseErrors

Entity - Response Errors

> **Note for API version 8.0**
>
> **As of API version 8.0**, Tachyon implements changes in the accessibility of instructions and their responses with the goal of enabling delegation of administration. The delegated user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. From the point of view of the Responses or Response Error APIs, this implies that the user is not allowed to see the responses for any instructions whose set of results might *potentially* contain responses from endpoints that should be invisible to the user. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but an error will be returned if the specified instruction is not accessible to the invoking user. Emphasis is done on "potentially", meaning that the system does not check whether the set of responses actually contains any response coming from a "forbidden" device, since this would be too costly at runtime. Instead, the system checks which management groups are accessible to the user who sent the instruction, and if this list contains any management group not accessible to the user who queries the responses, then the request is rejected (even if no response was actually received from devices in the forbidden management groups or if the Scope of the instruction prevented such devices from responding).
>
> For this purpose, the system uses the **current** Management Groups assigned to each user, **not** the Management Groups that the users could access back at the time when the instruction was submitted.

These are most often returned when there are no responses from an agent. There may be other scenarios if an agent is an error state.

Verb	Request	Notes	Permissions required
POST	/Consumer/ResponseErrors/{instructionId}	Allows fetching error responses.	• Viewer • Questioner

Verb	Request	Notes	Permissions required

		Example Request
		`{ "Filter":null, "PageSize":20, "Start":"0;0" }`
		• Filter - filter expression. If **null** then no filter is applied.To learn about how to define filter check Using scope and filter expressions page.
		• Start - starting index to support pagination. For the first page it has to be 0;0 and for the subsequent page it has to be the value returned as **Range** property from previous page request. Please check the example response below.
		• Pagesize - number of results to fetch
		Example Response
		`{ "Range": "MTsy", "Responses": [ { ... } ] }`
			• Actioner
			• Approver
			Instruction's owner can also view responses.
			Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.

Verb	Request	Notes	Permissions required
POST	/Consumer/ResponseErrors/{instructionId}/Aggregate	Allows fetching error responses. **Example Request** `{ "Filter":null, "PageSize":20 }` • Filter - filter expression. If **null** then no filter is applied.To learn about how to define filter check Using scope and filter expressions page. • Pagesize - number of results to fetch	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.
POST	/Consumer/ResponseErrors/{instructionId}/UniqueFQDNs	Returns the list of distinct FQDNs that are present in the Errors table for the specified instruction. It is invoked with the same parameters as the "Aggregate" API above. Returns an array of strings containing the FQDNs.	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.

2.3.44 Responses

Entity - Response

Note for API version 8.0

As of API version 8.0, Tachyon implements changes in the accessibility of instructions and their responses with the goal of enabling delegation of administration. The delegated user is supposed to only manage a subset of endpoints, which are specified by means of Management Groups. The delegated administrator should only be able to see information related to his or her Management Groups, and not anything else in the organization. From the point of view of the Responses API, this implies that the user is not allowed to see the responses for any instructions whose set of results might *potentially* contain responses from endpoints that should be invisible to the user. This is done transparently by the APIs unless otherwise noted, i.e., the API is called in the normal way as in previous versions, but an error will be returned if the specified instruction is not accessible to the invoking user. Emphasis is done on "potentially", meaning that the system does not check whether the set of responses actually contains any response coming from a "forbidden" device, since this would be too costly at runtime. Instead, the system checks which management groups are accessible to the user who sent the instruction, and if this list contains any management group not accessible to the user who queries the responses, then the request is rejected (even if no response was actually received from devices in the forbidden management groups or if the Scope of the instruction prevented such devices from responding).

For this purpose, the system uses the **current** Management Groups assigned to each user, **not** the Management Groups that the users could access back at the time when the instruction was submitted.

All of these return a similar collection.

Verb	Request	Notes	Permissions required
POST	/Consumer/Responses/{instructionId}	Allows fetching successful responses based on criteria. **Example Request** `{ "Filter":{ "Operator":"and", "Operands":[ { "Attribute":"Caption", "Operator":"==", "Value":"PLDS DVD+-RW DS-8A8SH" }, {`	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on

Verb	Request	Notes	Permissions required

		<pre>"Attribute":"Description",
		"Operator":"like",
		"Value":"Drive"
		}
		]
		},
		"PageSize":20,
		"Start":"0;0"
		}</pre>
		• **Filter** - filter expression. If **null** then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page.
		• **Pagesize** - number of results to fetch.
		• **Start** - starting index to support pagination. For the first page it has to be unencoded 0;0 and for the subsequent page it has to be the value returned as the encoded **Range** value passed back by the previous page request. You will get a base64 encoded string containing something like “1;4,2;2” which is a comma separated list
			the parent InstructionDefinition's product pack or is not the owner.

Verb	Request	Notes	Permissions required
		of pairs of ShardId;ResponseId. **Example Response** `{ "Range": "MTsy", "Responses": [{ "Id": 1, "ShardId": 1, "TachyonGuid": "00000000-0000-0000-0000-000000000000", "Fqdn": "machine01.fqdn.yourdomain", "ExecTime": 25000, "CreatedTimestampUtc": "2017-09-07T10:06:43.203Z", "ResponseTimestampUtc": "2017-09-07T10:06:41Z", "Values": { "FirstSchemaField": "someValue", "SecondSchemaField": "another value", ... }, "Blob": null }, { ... }]}`	
POST	/Consumer/Responses/{instructionId}/ Aggregate	Allows fetching aggregated successful responses. Only valid for instructions with aggregation capability dictated by the presence of aggregation schema.	• Viewer • Questioner • Actioner • Approver

Verb	Request	Notes	Permissions required
		Payload should be like the one below. **Example Request** `{ "Filter":{ "Operator":"and", "Operands":[ { "Attribute":"Caption", "Operator":"==", "Value":"PLDS DVD+-RW DS-8A8SH" }, { "Attribute":"Description", "Operator":"like", "Value":"Drive" } ] }, "PageSize":20 }` • **Filter** - filter expression. If **null** then no filter is applied. Searchable columns depends of schema. To learn about how to define filter check Using scope and filter expressions page. • **Pagesize** - number of results to fetch	Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.

Verb	Request	Notes	Permissions required
POST	/Consumer/Responses/{instructionId}/ UniqueFQDNs	Returns the list of distinct FQDNs that are present in the Responses table for the specified instruction. Will fail if invoked on an aggregated instruction that was not configured to keep the Raw data, since the aggregated data does not contain FQDNs. It is invoked with the same parameters as the "Aggregate" API above. Returns an array of strings containing the FQDNs.	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.
GET	/Consumer/Responses/Processed/{instructionId}	This endpoint is only usable for instructions that have Custom visualization(s) defined. This endpoint returns response data processed according to defined Custom visualizations and ready to be consumed by whatever visualization engine is to be used. Return data will vary depending on what visualizations the instruction uses.	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinitio

Verb	Request	Notes	Permissions required
			n's product pack or is not the owner.
GET	/Consumer/Responses/{instructionId:int}/ Shard/{shardId:int}/Row/{rowId:int}/ Column/{columnName}	Returns the value of a single column in a specific response. Designed to work with CLOB fields but will work with any data type. • instructionId - id of the instruction. This is the same value as the one you'd provide to Responses/{instructionId} endpoint • shardId - id of the shard the response is stored in. This value is provided inside the Response object returned by API when asking for non-aggregate responses as ShardId property (see example response above). • rowId - id of the row. This value is provided inside the Response object returned by API when asking for non-aggregate responses as Id property (see example response above).	• Viewer • Questioner • Actioner • Approver Instruction's owner can also view responses. Returns Unauthorized (401) if user does not have the required permissions at global level or on the parent InstructionDefinition's product pack or is not the owner.

Verb	Request	Notes	Permissions required
		• columnName - name of the column in the schema you want the value of. This has to match a column in the instruction definition's schema.	

2.3.45 ResponseTemplates

Entity - ResponseTemplates - handles Response Templates

Verb	Request	Notes	Permissions required
GET	/Consumer/ResponseTemplates	Returns all Response Templates within the system.	None required
GET	/Consumer/ResponseTemplates/Id/{id}	Returns a single Response Template.	None required

2.3.46 Role

Entity - Role Based Access Control - Interrogates the Roles table in the database

Verb	Request	Notes	Permissions required
GET - all	/Consumer/Roles	Gets list of all Roles	• Read Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required

GET - single	/Consumer/Roles/{roleId}	Gets details for the specified Role ID.
		
		Example Response
		
		{
		"Id": 1,
		"Name": "Administrators",
		"Description": "This role has permissions to everything",
		"CreatedTimestampUtc": "2015-09-15T09:45:58.73",
		"ModifiedTimestampUtc": "2015-09-15T09:45:58.73"
		}
			• Read
			
			Returns Unauthorized (401) if user does not have the required permissions.

GET - for a principal	/Consumer/Roles/Principal/{principalId}	Gets all roles for a specific principal **Obsolete as of API version 8.0** . Retained for backwards compatibility. Will work as suggested /PrincipalRoleManagementGroup equivalent.	• Read Returns Unauthorized (401) if user does not have the required permissions.

POST	/Consumer/Roles/Search	**Minimum API version 4.0**
		
		This endpoint allows searching for roles based on criteria along with sorting and pagination.
		
		Example request
		
		{
		"filter":{
		"Attribute":"Name",
		"Operator":"LIKE",
		"Value":"%Administrator%"
		},
			Requires 'Read' on 'Security'

Verb	Request	Notes	Permissions required

		<pre>"start":10,
		"pageSize":100,
		"sort":[
		{
		"Column":"SystemRole",
		"Direction":"DESC"
		}
		]
		}</pre>	

POST	/Consumer/Roles/SearchWithCounts	**Minimum API version 8.0**
		
		Works in the same way as /Consumer/Roles/Search discussed above, but the returned results include additional fields that indicate the usage count for each returned role. The counts provide the number of Instruction Sets assigned to the role as well as the number of Securable Types on which the Role has one or more permissions, differentiating between local and global securable types.
		
		Sample output
		
		<pre>{
		"TotalCount": 37,
		"Items": [
		{
		"InstructionSets": 0,
		"LocalizedOperations": 0,
		"GlobalPermissions": 1,
		"Id": 11,
		"Name": "1E Client Deployment Administrators",
		"Description":</pre>
			Requires 'Read' on 'Security'

Verb	Request	Notes	Permissions required

		<pre>"Create, view and cancel 1E Client deployment jobs",
		"CreatedTimestampUtc": "2020-02-21T09:23:34.047Z",
		"ModifiedTimestampUtc": "2020-04-14T15:30:02.953Z",
		"SystemRole": true,
		"CanBeDelegated": false,
		"NumberOfAssignments": 0
		},
		...
		{
		"InstructionSets": 0,
		"LocalizedOperations": 2,
		"GlobalPermissions": 0,
		"Id": 33,
		"Name": "Survey Viewers",
		"Description": "View surveys",
		"CreatedTimestampUtc": "2020-05-11T09:53:24.4Z",
		"ModifiedTimestampUtc": "2020-05-11T09:53:24.4Z",
		"SystemRole": false,
		"CanBeDelegated": false,
		"NumberOfAssignments": 0
		}
		]
		}</pre>
		Note that in version 8.0, the "Global Adminstrators" role has become a pseudo-role that	

Verb	Request	Notes	Permissions required
		automatically receives all possible permissions. The API as discussed above only returns the count of permissions that are explicitly assigned in the database, so it will not count the implicit permissions that are internally granted to the pseudo-role.	
POST	/Consumer/Roles	Creates a Role record. **Example Request** `{ "Name": "Administrators", "Description": "This role has permissions to everything" }`	• Write Returns Unauthorized (401) if user does not have the required permissions.
PUT	/Consumer/Roles	Modifies an existing Role. **Example Request** `{ "Id": 1, "Name": "Administrators", "Description": "This role has permissions to everything" }`	• Write Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/Roles/{roleId}	Deletes the specified Role if it exists, otherwise "ExceptionMessage": "Role record with Id={roleId} was not found".	• Delete Returns Unauthorized (401) if user does not have

Verb	Request	Notes	Permissions required
			the required permissions.
DELETE	/Consumer/Roles	**Minimum API version 4.0** Deletes multiple roles. System roles cannot be deleted. Body or the request should contain an array of Role Ids.	Requires 'Delete' on Security
GET	/Consumer/Roles/{id}/ManagementGroups	**Minimum API version 3.3** Returns all Management Groups given role has permissions to access **Removed as of API version 8.0** . Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Roles through Permissions.	Requires 'Read' on Security
POST	/Consumer/Roles/{roleId}/ManagementGroups/{managementGroupId}	**Minimum API version 3.3** Adds permission to use given Management Group to given Role. • roleId - Id of the Role to which the permission should be granted. • managementGroupId - Id of the Management Group to which the permission in granted.	Requires 'Write' on Security

Verb	Request	Notes	Permissions required

		Return payload will contain the Role in question and all Management Groups that Role has permissions on, not only the ones that have been added.
		
		Sample Response payload Expand source
		
		{
		"Role": {
		"Id": 1,
		"Name": "Administrators",
		"Description": "This role has permissions to everything",
		"CreatedTimestampUtc": "2015-09-15T09:45:58.73",
		"ModifiedTimestampUtc": "2015-09-15T09:45:58.73"
		},
		"ManagementGroups": [
		{
		"Id": "1",
		"Name": "Management Group A",
		"Description": "",
		"Expression": "",
		"Count": "12",
		"UsableId": "0E9514B6-FD77-4077-9B54-A2AAA0C1953C",
		"HashOfMembers": "5c9f347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5b754"
		},{
		"Id": "2",
		"Name": "Ma	

Verb	Request	Notes	Permissions required
		nagement Group B", "Description": "", "Expression": "", "Count": "7", "UsableId": "AD9514B6-FD77-4077-9B54-A2AAA0C1953C", "HashOfMembers": "ddff347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5bbcc" }] } **Removed as of API version 8.0** . Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Roles through Permissions.	

POST	/Consumer/Roles/ManagementGroups	**Minimum API version 3.3**
		
		Adds permission to use specified Management Groups to given Role.
		
		Sample Request payload
		
		{
		"RoleId": 1,
		"ManagementGroupIds": [1,2,3]
		}
		
		Return payload will contain the Role in question and all Management Groups that Role
			Requires 'Write' on Security

Verb	Request	Notes	Permissions required

		has permissions on, not only the ones that have been added.
		
		Sample Response payload Expand source
		
		{
		"Role": {
		"Id": 1,
		"Name": "Administrators",
		"Description": "This role has permissions to everything",
		"CreatedTimestampUtc": "2015-09-15T09:45:58.73",
		"ModifiedTimestampUtc": "2015-09-15T09:45:58.73"
		},
		"ManagementGroups": [
		{
		"Id": "1",
		"Name": "Management Group A",
		"Description": "",
		"Expression": "",
		"Count": "12",
		"UsableId": "0E9514B6-FD77-4077-9B54-A2AAA0C1953C",
		"HashOfMembers": "5c9f347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5b754"
		},{
		"Id": "2",
		"Name": "Management Group B",
		"Description": "",
		"Expression	

Verb	Request	Notes	Permissions required

		<pre>": "",
		"Count": "7",
		"UsableId": "AD9514B6-FD77-4077-9B54-A2AAA0C1953C",
		"HashOfMembers": "ddff347457272d5edfeb5dd5c0f6fe4e38311b6126908f2612a1e58d8fb5bbcc"
		}
		]
		
		}</pre> **Removed as of API version 8.0** . Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Roles through Permissions.	

DELETE	/Consumer/Roles/{roleId}/ManagementGroups/{managementGroupId}	**Minimum API version 3.3** Removes permission to use given Management Group to given Role. • roleId - Id of the Role from which the permission should be removed. • managementGroupId - Id of the Management Group to which the permission in removed. Return payload is empty. **Removed as of API version 8.0** . Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because	Requires 'Write' on Security

Verb	Request	Notes	Permissions required
		Management Groups are no longer linked to Roles through Permissions.	

DELETE	/Consumer/Roles/ManagementGroups	**Minimum API version 3.3**
		
		Removes the permission to use specified Management Groups form given Role.
		
		Sample Request payload
		
		{
		"RoleId": 1,
		"ManagementGroupIds": [1,2,3]
		}
		
		Return payload is empty.
		
		Removed as of API version 8.0 . Will return 405 if called. It cannot work anymore or be made backwards compatible in a sensible way because Management Groups are no longer linked to Roles through Permissions.
			Requires 'Write' on Security

POST	/Consumer/Roles/Complete	Creates a new role and assigns given permissions and management groups to it
		
		Sample request Expand source
		
		{
		"Name": "Complete Role 1",
		"Description": "This is a test role",
		"Permissions": [{
		"SecurableId": 2,
		"SecurableTypeId": 1,
		"Allowed": true,
			Requires 'Write' on Security

Verb	Request	Notes	Permissions required


```
            "Operations": [{
                "OperationId":
2
            },
            {
                "OperationId":
4
            },
            {
                "OperationId":
3
            },
            {
                "OperationId":
1
            }]
        },
        {
            "SecurableId": 1,
            "SecurableTypeId":
1,
            "Allowed": true,
            "Operations": [{
                "OperationId":
1
            },
            {
                "OperationId":
4
            }]
        }],
        "ManagementGroupIds":
[10,11,12]
}
```


Sample response Expand source


```
{
    "Role": {
        "Id": 1018,
        "Name": "Complete R
ole 1",
        "Description": "Thi
s is a test role",
        "CreatedTimestampUt
c": "2018-07-25T07:24:50.01
39416Z",
```

Verb	Request	Notes	Permissions required


```
        "ModifiedTimestampUtc": "2018-07-25T07:24:50.0139416Z",
        "SystemRole": false
    },
    "Permissions": [{
        "SecurableId": 1,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1018,
        "RoleName": "Complete Role 1",
        "Allowed": true,
        "Operations": [{
            "PermissionId": 1074,
            "OperationId": 1,
            "OperationName": "Viewer",
            "CreatedTimestampUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        },
        {
            "PermissionId": 1075,
            "OperationId": 4,
            "OperationName": "Approver",
            "CreatedTimestampUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        }]
    },
    {
```

Verb	Request	Notes	Permissions required


```
        "SecurableId": 2,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1018,
        "RoleName": "Complete Role 1",
        "Allowed": true,
        "Operations": [{
            "PermissionId": 1070,
            "OperationId": 2,
            "OperationName": "Actioner",
            "CreatedTimestampUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        },
        {
            "PermissionId": 1071,
            "OperationId": 4,
            "OperationName": "Approver",
            "CreatedTimestampUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        },
        {
            "PermissionId": 1072,
            "OperationId": 3,
            "OperationName": "Questioner",
            "CreatedTimesta
```

Verb	Request	Notes	Permissions required


```
mpUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        },
        {
            "PermissionId": 1073,
            "OperationId": 1,
            "OperationName": "Viewer",
            "CreatedTimestampUtc": "2018-07-25T07:24:50.057Z",
            "ModifiedTimestampUtc": "2018-07-25T07:24:50.057Z"
        }]
    }],
    "ManagementGroups": [{
        "Id": 10,
        "Name": "Management group number 9",
        "Description": "This is the description for Management group number 9",
        "Expression": "ADSiteName==Site9",
        "Type": 2,
        "Count": 0,
        "UsableId": "Id-9",
        "HashOfMembers": "29B9C10CF281F598621A069E8B660AE0D518822C870F94CA4D28FBA7F04EC944"
    },
    {
        "Id": 11,
        "Name": "Management group number 10",
        "Description": "This is the description for Management group number 10",
        "Expression": "ADSiteName==Site10",
```

Verb	Request	Notes	Permissions required

		<pre>"Type": 2,
		"Count": 0,
		"UsableId": "Id-10
		",
		"HashOfMembers": "8
		03B7F61B5FFD589FA88DD89527A
		0EA6C32BD1C18C8B3796B26D110
		AECF6AC12"
		},
		{
		"Id": 12,
		"Name": "Management
		group number 11",
		"Description": "Thi
		s is the description for Ma
		nagement group number 11",
		"Expression": "ADSi
		teName==Site11",
		"Type": 2,
		"Count": 0,
		"UsableId": "Id-11
		",
		"HashOfMembers": "B
		A40925847BA8F51113AE9ADE326
		7F4EF9696BB799B6EBA41FB3376
		993664DAE"
		}]
		}</pre> **Behaviour changed as of API version 8.0** . The payload sent to the endpoint and returned from it will no longer have Management Group related information because Management Groups are not linked directly to Roles but require a Principal as well. From v8.0 this endpoint can still be used to create a Role with assigned Permissions, but it cannot be used to also link a Management Group to that role, because that type of link no longer exists.	

Verb	Request	Notes	Permissions required

PUT	/Consumer/Roles/Complete	Updates a role and assigns given permissions and management groups.
		
		Any previous permissions are removed.
		
		Any previously linked management groups are unlinked.
		
		Group after update will only have permissions and management groups specified in the request.
		
		Sample request Expand source
		
		{
		"Id": 1018,
		"Name": "Complete Role 1",
		"Description": "Updated",
		"Permissions": [{
		"SecurableId": 2,
		"SecurableTypeId": 1,
		"Allowed": true,
		"Operations": [{
		"OperationId": 1
		}]
		},
		{
		"SecurableId": 1,
		"SecurableTypeId": 1,
		"Allowed": true,
		"Operations": [{
		"OperationId": 1
		},
		{
		"OperationId": 2
		},
		{
			Requires 'Write' on Security

Verb	Request	Notes	Permissions required


```
            "OperationId":
3
        },
        {
            "OperationId":
4
        }]
    }],
    "ManagementGroupIds":
[10,20,21]
}
```


Sample response Expand source


```
{
    "Role": {
        "Id": 1018,
        "Name": "Complete R
ole 1",
        "Description": "Thi
s is a role",
        "CreatedTimestampUt
c": "2018-07-25T07:24:50.01
3Z",
        "ModifiedTimestampU
tc": "2018-07-25T13:46:20.8
597827Z",
        "SystemRole": false
    },
    "Permissions": [{
        "SecurableId": 1,
        "SecurableName": nu
ll,
        "SecurableTypeId":
1,
        "SecurableTypeName
": "InstructionSet",
        "RoleId": 1018,
        "RoleName": "Comple
te Role 1",
        "Allowed": true,
        "Operations": [{
            "PermissionId":
1143,
            "OperationId":
1,
            "OperationName
```

Verb	Request	Notes	Permissions required


```
": "Viewer",
            "CreatedTimestampUtc": "2018-07-25T13:46:20.907Z",
            "ModifiedTimestampUtc": "2018-07-25T13:46:20.907Z"
        },
        {
            "PermissionId": 1144,
            "OperationId": 2,
            "OperationName": "Actioner",
            "CreatedTimestampUtc": "2018-07-25T13:46:20.907Z",
            "ModifiedTimestampUtc": "2018-07-25T13:46:20.907Z"
        },
        {
            "PermissionId": 1145,
            "OperationId": 3,
            "OperationName": "Questioner",
            "CreatedTimestampUtc": "2018-07-25T13:46:20.907Z",
            "ModifiedTimestampUtc": "2018-07-25T13:46:20.907Z"
        },
        {
            "PermissionId": 1146,
            "OperationId": 4,
            "OperationName": "Approver",
            "CreatedTimestampUtc": "2018-07-25T13:46:20.907Z",
            "ModifiedTimest
```

Verb	Request	Notes	Permissions required


```
ampUtc": "2018-07-25T13:46:
20.907Z"
        }]
    },
    {
        "SecurableId": 2,
        "SecurableName": nu
ll,
        "SecurableTypeId":
1,
        "SecurableTypeName
": "InstructionSet",
        "RoleId": 1018,
        "RoleName": "Comple
te Role 1",
        "Allowed": true,
        "Operations": [{
            "PermissionId":
1142,
            "OperationId":
1,
            "OperationName
": "Viewer",
            "CreatedTimesta
mpUtc": "2018-07-25T13:46:2
0.907Z",
            "ModifiedTimest
ampUtc": "2018-07-25T13:46:
20.907Z"
        }]
    }],
    "ManagementGroups": [{
        "Id": 10,
        "Name": "Management
group number 9",
        "Description": "Thi
s is the description for Ma
nagement group number 9",
        "Expression": "ADSi
teName==Site9",
        "Type": 2,
        "Count": 0,
        "UsableId": "Id-9",
        "HashOfMembers": "2
9B9C10CF281F598621A069E8B66
0AE0D518822C870F94CA4D28FBA
7F04EC944"
```

Verb	Request	Notes	Permissions required

		<pre> },
		    {
		        "Id": 20,
		        "Name": "Management group number 19",
		        "Description": "This is the description for Management group number 19",
		        "Expression": "ADSiteName==Site19",
		        "Type": 2,
		        "Count": 0,
		        "UsableId": "Id-19",
		        "HashOfMembers": "C1F521A70A60FB5BB3EC348D25D8E4B37DB6844428F9A69029D73BE140887C2B"
		    },
		    {
		        "Id": 21,
		        "Name": "Management group number 20",
		        "Description": "This is the description for Management group number 20",
		        "Expression": "ADSiteName==Site20",
		        "Type": 2,
		        "Count": 0,
		        "UsableId": "Id-20",
		        "HashOfMembers": "BF92D8BA0315CF4C60FDBF0DACE7C864E5C3062D654A3FB25C434791D380F60C"
		    }]
		}</pre>
		Behaviour changed as of API version 8.0 . The payload sent to the endpoint and returned from it will no longer have Management Group related information because Management Groups are not linked directly to Roles but	

Verb	Request	Notes	Permissions required
		require a Principal as well. From v8.0 this endpoint can still be used to update a Role with assigned Permissions, but it cannot be used to also link a Management Group to that role, because that type of link no longer exists.	

2.3.47 Rules

Entiry - Rules

Minimum API version 4.0

Verb	Request	Notes	Permissions Required
GET	/Consumer/Rules/{id}	Returns a specific Rule	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/Rules	Returns all Rules	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/Rules/Search	Returns Rules that match the search parameters specified. • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria	Requires 'Read' permission to 'GuaranteedState' securable type

Verb	Request	Notes	Permissions Required
		check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Name • Type • ReadablePayload • Enabled Allowed sort columns: • Name • Type • ReadablePayload • Enabled	
POST	/Consumer/Rules/Shallow/Search	Returns Rules that match the search parameters specified. This endpoint returns only superficial information about Fragments used by the Rule. For detailed infirmation please use /Rules/Search instead. • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page.	Requires 'Read' permission to 'GuaranteedState' securable type

Verb	Request	Notes	Permissions Required
		• sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Name • Type • ReadablePayload • Enabled Allowed sort columns: • Name • Type • ReadablePayload • Enabled	
POST	/Consumer/Rules	Creates a new Rule	Requires 'Write' permission to 'GuaranteedState' securable type
DELETE	/Consumer/Rules/{id}	Deletes a Rule	Requires 'Delete' permission on 'GuaranteedState' securable type

Verb	Request	Notes	Permissions Required
POST	/Consumer/Rules/{id}/Enable/{enable}	If 'enable' is set to True it enables given Rule. If it is set to False the Rule will be disabled.	Requires 'Write' permission to 'GuaranteedState' securable type
GET	/Consumer/Rules/Unlicensed	Minimum API version 5.1 Retrieves a list of Ids of Rules that use unlicensed Fragments	Requires 'Read' permission on 'GuaranteedState' securable type

2.3.48 ScheduledInstructions

Entity - ScheduledInstructions - Interrogates the ScheduledInstruction table

Note for API version 8.0

As of API version 8.0, the visibility of the Scheduled Instructions is restricted in accordance to the Management Groups assigned to the users. See the note at the top of the Instructions page for an explanation of how this works.

Verb	Request	Notes	Permissions required
GET	Consumer/ScheduledInstructions/	Gets all scheduled instructions Note: For an explanation of the possible values of the ScheduleXxx parameters, see https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/dbo-sysschedules-transact-sql Not every possible combination of values for these parameters will be initially implemented in Tachyon. **Returned JSON** [{	None

Verb	Request	Notes	Permissions required


```
    "Id": 4,
    "Name": "Stable-Coun
tEventLogErrorsInLastX",
    "Cmd": "SendAll",
    "KeepRaw": false,
    "Scope": null,
    "ReadableScope": nul
l,
    "InstructionTtlMinut
es": 10,
    "ResponseTtlMinutes
": 10,
    "Export": false,
    "ExportLocation": nu
ll,
    "ParentInstructionId
": null,
    "InstructionDefiniti
onId": 42347,
    "CreatedBy": "1E\\Al
berto.Poblacion",
    "ResultsFilter": nul
l,
    "PreviousResultsFilt
er": null,
    "ConsumerId": 1,
    "ConsumerName": "Exp
lorer",
    "ConsumerCustomData
": null,
    "ParameterJson": "
[{\"Name\":\"type\",\"Pa
ttern\":\"%type%\",\"Dat
aType\": ... }]",
    "OffloadResponses":
false,
    "RequestedFor": nul
l,
    "Comments": "Just te
sting from PostMan",
    "ActionedBy": null,
    "ActionReason": nul
l,
    "ScheduleEnabled": t
rue,
    "ScheduleFreqType":
4,
```

Verb	Request	Notes	Permissions required

		<pre>"ScheduleFreqInterval": 1,
		"ScheduleFreqSubdayType": 1,
		"ScheduleFreqSubdayInterval": 1,
		"ScheduleFreqRelativeInterval": 0,
		"ScheduleFreqRecurrenceFactor": 0,
		"ScheduleActiveStartDate": null,
		"ScheduleActiveEndDate": null,
		"ScheduleActiveStartTime": null,
		"ScheduleActiveEndTime": null,
		"ScheduleDateCreated": "2017-05-09T14:53:51.19Z",
		"ScheduleDateModified": "2017-05-09T14:53:51.157Z",
		"ScheduleLastExecuted": null,
		"ScheduleNextExecution": "2017-05-10T14:53:51.163Z"
		}
		]	

GET - single	/Consumer/ScheduledInstructions/ {id}	Gets an individual row from the ScheduledInstruction table by its Id.
		
		Example Response
		
		<pre>{
		"Id": 4,
		"Name": "Stable-CountEventLogErrorsInLastX",
		"Cmd": "SendAll",
		"KeepRaw": false,
		"Scope": null,
		"ReadableScope": null,</pre>
			• Viewer
			• Questioner
			• Actioner
			• Approver
			
			Returns Unauthorized (401) if user does not have the required permissions at global level or on

Verb	Request	Notes	Permissions required

		<pre>"InstructionTtlMinutes": 10,
		"ResponseTtlMinutes": 10,
		"Export": false,
		"ExportLocation": null,
		"ParentInstructionId": null,
		"InstructionDefinitionId": 42347,
		"CreatedBy": "1E\\Alberto.Poblacion",
		"ResultsFilter": null,
		"PreviousResultsFilter": null,
		"ConsumerId": 1,
		"ConsumerName": "Explorer",
		"ConsumerCustomData": null,
		"ParameterJson": "[{\"Name\":\"type\",\"Pattern\":\"%type%\",\"DataType\":\"string\", ... }]",
		"OffloadResponses": false,
		"RequestedFor": null,
		"Comments": "Just testing from PostMan",
		"ActionedBy": null,
		"ActionReason": null,
		"ScheduleEnabled": true,
		"ScheduleFreqType": 4,
		"ScheduleFreqInterval": 1,
		"ScheduleFreqSubdayType": 1,
		"ScheduleFreqSubdayInterval": 1,
		"ScheduleFreqRelativeInterval": 0,
		"ScheduleFreqRecurrenceFactor": 0,
		"ScheduleActiveStartDate": null,</pre>
			the parent InstructionDefinition's product pack.

Verb	Request	Notes	Permissions required

		<pre>"ScheduleActiveEndDate": null,
		"ScheduleActiveStartTime": null,
		"ScheduleActiveEndTime": null,
		"ScheduleDateCreated": "2017-05-09T14:53:51.19Z",
		"ScheduleDateModified": "2017-05-09T14:53:51.157Z",
		"ScheduleLastExecuted": null,
		"ScheduleNextExecution": "2017-05-10T14:53:51.163Z"
		}</pre>	

POST	/Consumer/ScheduledInstructions	Creates a scheduled question, action or event.
		
		Example Request
		
		<pre>{
		"DefinitionName":"TachyonCore-BiosDetails",
		"InstructionTtlMinutes":10,
		"ResponseTtlMinutes":10,
		"Comments":"This is a comment",
		"ScheduleEnabled": true,
		"ScheduleFreqType": 4,
		"ScheduleFreqInterval": 1,
		"ScheduleFreqSubdayType": 1,
		"ScheduleFreqSubdayInterval": 1,
		"ScheduleFreqRelativeInterval": 0,
		"ScheduleFreqRecurrenceFactor": 0,</pre>
			For questions
			• Questioner
			• Actioner
			For actions
			• Actioner
			Returns Unauthorized (401) if user does not have the required permissions on the instruction that is being scheduled.
			
			NOTE: The attached JSON is just an example to demonstrate how to pass the Scheduling parameters.

Verb	Request	Notes	Permissions required
		"ScheduleDateCreated": "2017-05-09T14:50:01.343Z", "ScheduleDateModified": "2017-05-09T14:50:01.343Z" } will return: **Example Response** [Similar to GET]	Otherwise, it accepts parameters that are similar to those allowed by the /Consumer/Instruction API.
PUT	/Consumer/ScheduledInstructions/ Update/{id}	Takes a ScheduledInstruction (using the same schema as POST) and replaces it into the existing ScheduledInstruction whose Id is given by {id}.	Permissions as above.
POST	/Consumer/ScheduledInstructions/ Targeted	Creates a scheduled instruction, supplying a targeted list of devices by Fqdn As above with the added "Devices" Json property e.g. **Example Request** { "DefinitionName":"TachyonCore-BiosDetails", "InstructionTtlMinutes":10, "ResponseTtlMinutes":10, "Comment":"Just testing from PostMan", "ScheduleEnabled": true, "ScheduleFreqType": 4, "ScheduleFreqInterval": 1, "ScheduleFreqSubdayType	Permissions as above. Note: Fqdn list is limited by MaxRequestLength which defaults to 4MB

Verb	Request	Notes	Permissions required

		<pre>": 1,
		"ScheduleFreqSubdayInterval": 1,
		"ScheduleFreqRelativeInterval": 0,
		"ScheduleFreqRecurrenceFactor": 0,
		"ScheduleDateCreated": "2017-05-08T11:43:01.343Z",
		"ScheduleDateModified": "2017-05-08T11:43:01.343Z",
		"Devices":["1EUKDEVWKS1349.1e.local"]
		}</pre>	

PUT	/Consumer/ScheduledInstructions/ Targeted/{id}	Takes a ScheduledInstruction (using the same schema as POST, including list of devices by Fqdn) and replaces it into the existing ScheduledInstruction whose Id is given by {id}.	Permissions as above.
DELETE	/Consumer/ScheduledInstructions/{id}	Deletes an individual row from the ShcheduledInstruction table by its Id.	Permission to execute schedule
POST	/Consumer/ScheduledInstructions/ {id}/cancel	Cancel an individual schedule	Permission to execute schedule

POST	/Consumer/ScheduledInstructions/ search	Find all occurrences of a schedule between two dates.
		Example Request
		<pre>{
		"Filter":{
		"Operator": "AND",
		"Operands": [
		{</pre>	

Verb	Request	Notes	Permissions required

		<pre>"Attribute": "DefinitionName",
		"Operator": "==",
		"Value": "TachyonCore-BiosDetails"
		}
		]
		},
		"StartDate": "2017-05-01T00:00:00+0100",
		"EndDate": "2017-07-01T00:00:00+0100",
		"PageSize": 20
		}</pre>
		will return:
		Example Response
		<pre>[Similar to GET: An array of ScheduleDefinitions where the relevant information is the ScheduleNextExecution field]</pre>
		For past executions, the response contains an InstructionId field that reports the ID from the instruction that was triggered from the schedule. This field is null for future executions, since it is not possible to predict the ID that will be generated when the schedule is triggered.
		Note: Either **EndDate** or **PageSize** needs to be specified when invoking this API. StartDate is optional, and it defaults to "the first	

Verb	Request	Notes	Permissions required
		occurrence of this schedule that ever happened".	
POST	/Consumer/ScheduledInstructions/ Describe	**Minimum API version - 3.2** Returns the description for a Scheduled Instruction even if it is not saved into the system. Takes as input parameter a ScheduledInstruction, similar to what would be sent to /Consumer/ScheduledInstructions in order to add the schedule to the system. This API does not do anything with the schedule, except populate some data inside the object, namely the ScheduleReadableFrequency. Returns the same object, including the newly populated information.	None requried.
GET	/Consumer/ScheduledInstructions/ approxtarget	**Minimum API version - 3.3** Gets approximate target for specific instruction. Approximate target will be calculated using the scope defined in the instruction. Approximate target will be calculated for the user who created the instruction, using that user's current permissions. **New for API version 8.0**: The calculation for the	Requires any permission on any of the instruction sets

Verb	Request	Notes	Permissions required
		approximate target only counts the Devices that the current user is allowed to see in accordance to the Management Groups that are assigned to the user.	

2.3.49 SecurableTypes

Entity - Role Based Access Control - Interrogates the SecurableType table in the database

Verb	Request	Notes	Permissions required
GET - all	/Consumer/SecurableTypes	Gets list of all available Securable Types The fields *IsGlobal* and *Description* are new for version 8.0. **Example Response** [{ "Id": 1, "Name": "ProductPack", "CreatedTimestampUtc": "2016-02-03T17:20:06.927", "ModifiedTimestampUtc": "2016-02-03T17:20:06.927", "ApplicableOperations": [], "Operations": [{ "Id":1, "OperationName":"Viewer", "SecurableTypeId":1, "Securab	• Read Returns Unauthorized (401) if user does not have the required permissions.

Verb	Request	Notes	Permissions required


```
leTypeName":"ProductPack"
                },
                {
                    "Id":2,
                    "Operati
onName":"Actioner",
                    "Securab
leTypeId":1,
                    "Securab
leTypeName":"ProductPack"
                }
            ],
            "IsGlobal": fals
e,
            "Description": "P
roduct Pack"
        },
        {
            "Id": 2,
            "Name": "Security
",
            "CreatedTimestamp
Utc": "2016-02-03T17:20:0
6.927",
            "ModifiedTimestam
pUtc": "2016-02-03T17:20:
06.927",
            "ApplicableOperat
ions": [],
            "Operations": [
                {
                    "Id": 5,
                    "Operati
onName": "Read",
                    "Securab
leTypeId": 2,
                    "Securab
leTypeName": "Security"
                }
            ],
            "IsGlobal": true,
            "Description": "S
ecurity"
        },
]
```

Verb	Request	Notes	Permissions required
GET - single	/Consumer/SecurableTypes/{securableId}	Gets details for the specified Securable ID **Example Request** `{ "Id": 1, "Name": "ProductPack", "CreatedTimestampUtc": "2016-02-03T17:20:06.927", "ModifiedTimestampUtc": "2016-02-03T17:20:06.927", "ApplicableOperations": [], "Operations": [{ "Id":2, "OperationName":"Actioner", "SecurableTypeId":1, "SecurableTypeName":"ProductPack" }, { "Id":1, "OperationName":"Viewer", "SecurableTypeId":1, "SecurableTypeName":"ProductPack" }], "IsGlobal": false, "Description": "Product Pack" }`	• Read Returns Unauthorized (401) if user does not have the required permissions.
GET	/Consumer/SecurableTypes/Name/{name}	As above but by name	• Read Returns Unauthorized (401)

Verb	Request	Notes	Permissions required
			if user does not have the required permissions.
POST	/Consumer/SecurableTypes	Creates a securable type record. The *Description* field is new for version 8.0. **Example Request** `{ "Name": "SecurableType", "Description": "Some text" }`	• Write Returns Unauthorized (401) if user does not have the required permissions.
PUT	/Consumer/SecurableTypes	Modifies an existing securable type entry. Only Name and Description can be modified. **Example Request** `{ "Id": 10, "Name": "Modified Securable Type", "Description": "Some text" }`	• Write Returns Unauthorized (401) if user does not have the required permissions.
DELETE	/Consumer/SecurableTypes/{securableId}	Deletes the specified securable type if it exists, otherwise "ExceptionMessage": "SecurableType record with Id={id} was not found".	• Write Returns Unauthorized (401) if user does not have the required permissions.

2.3.50 Settings

Entity - Global Settings - Interrogates Global Settings table

Verb	Request	Notes	Permissions Required
GET	/Consumer/Settings/{key}	Key must be a valid string from following list: • MaximumInstructionTtlMinutes • MaximumResponseTtlMinutes • MaximumSimultaneousInFlightInstructions Keys are case sensitive.	None

2.3.51 SystemInformation

Entity - System Information - Returns information about the system

Verb	Request	Notes	Permissions required

GET	/Consumer/SystemInformation	Returns a representation of the system's topology
		Example response
		{
		"ConsumerAPI": "http://localhost/Consumer/",
		"Coordinator": "http://localhost:8080/api/",
		"BackgroundChannels": [
		{
		"URL": "http://localhost/background/",
		"ClientCertificateAuthentication": false
		}
		],
		"Cores": [
		{
		"Name": "localhost",
			Read permission on Instrumentation
			In version 5.1 Instrumentation has been renamed to Infrastructure

Verb	Request	Notes	Permissions required


```
            "BaseURL":
"http://localhost/Core
/",
            "Response
": "http://localhost/C
ore/Responses",
            "Instructi
on": "http://localhost
/Core/Instructions",
            "Device":
"http://localhost/Core
/Devices",
            "Switch":
"http://localhost/Core
/Switches",
            "DataStore
s": [
                "(Loca
l to Core)"
            ]
        }
    ],
    "Switches": [
        {
            "Name": "T
achyon.acme.local",
            "Status":
"Online",
            "Listening
IpAddress": "127.0.0.1
",
            "Listening
Port": 4000,
            "PokeIpAdd
ress": "127.0.0.1",
            "ManagerPo
rt": 4001,
            "Workers":
4,
            "Slots": 3
2,
            "SSL": " T
LSv1.2",
            "SecurityL
evel": 1,
            "Cores": [
                "local
```

Verb	Request	Notes	Permissions required

		<pre>host"
		            ]
		        }
		    ]
		}</pre>	

GET	/Consumer/SystemInformation/GetSystemTopography	Returns a representation of the system's topology
		
		Example response
		
		<pre>{
		  "DataStores": [
		    {
		      "Id": 1,
		      "Name": "(local)
		",
		      "ProviderName":
		"System.Data.SqlClient
		",
		      "Cores": [
		        {
		          "Id": 1,
		          "Name": "loc
		alhost",
		          "Switches":
		[
		            {
		              "Id": 1,
		              "Name":
		"1EUKDEVWKS1204.1E.LOC
		AL",
		              "Slots":
		256,
		              "Agents
		": 0
		            },
		            {
		              "Id": 2,
		              "Name":
		"LAB-CM2012.Lab.local
		",
		              "Slots":
		256,
		              "Agents
		": 0
		            }</pre>
			Read permission on Instrumentation
			
			In version 5.1 Instrumentation has been renamed to Infrastructure

Verb	Request	Notes	Permissions required


```
          ]
        }
      ]
    },
    {
      "Id": 2,
      "Name": "lab-cm2
012",
      "ProviderName":
"System.Data.SqlClient
",
      "Cores": [
        {
          "Id": 2,
          "Name": "lab
-cm2012",
          "Switches":
[
            {
              "Id": 3,
              "Name":
"LAB-SERVER2.Lab.local
",
              "Slots":
256,
              "Agents
": 0
            }
          ]
        }
      ]
    },
    {
      "Id": 3,
      "Name": "lab-ser
ver2",
      "ProviderName":
"System.Data.SqlClient
",
      "Cores": [
        {
          "Id": 3,
          "Name": "lab
-server2",
          "Switches":
[]
        }
```

Verb	Request	Notes	Permissions required

		<pre>]
		}
		]
		}</pre>	

GET	/Consume/SystemInformation/License	Returns license information
		
		Return Json
		
		<pre>{
		"Customer": "Name of the customer",
		"CustomerEmail": "mrcustomer@somedomain.com",
		"CustomerTel": "0123456789",
		"LicenseSerialNumber": "123",
		"IssueDate": "2017/06/26",
		"ExpiryDate": "2017/07/26",
		"ActivationPeriodDays": "3",
		"ActivationGraceDays": "2",
		"Products": [{
		"Name": "Tachyon",
		"ExeNames": "Tachyon,Switch,Tachyon.Switch,Tachyon.Server.Core",
		"MaxUsers": "188",
		"Features": [
		{
		"Name": "NightWatchmanIntegration",
		"Expiry": "2017/06/28",
		"Instructions": [{
		"SignerS</pre>
			None

Verb	Request	Notes	Permissions required

		<pre>ha": "888888888888888",
		"Pattern": "1E-NWM-*"
		}],
		
		"Consumers": []
		},
		{
		"Name": "TachyonExplorer",
		"Expiry": "2017/06/30",
		
		"Instructions": [],
		"Consumers": [{
		"Name": "Explorer",
		"Enable": "off"
		}]
		}
		]
		}],
		"LicenseService": {
		"SignerCertName": "antec64.cer",
		"ActivateCertName": "antec64.cer",
		"Url": "https://license.1e.com",
		"AltUrl": "https://license.1e.local"
		}
		}</pre>	

GET	/Consumer/SystemInformation/LicenseExpiry/Product/{product}/Feature/{feature}	Returns expiration date for given product and feature
		
		Return JSON
		
		<pre>{
		"DaysLeft": 20,
		"ExpiryDateTim</pre>
			None

Verb	Request	Notes	Permissions required
		e": "2017/07/27 01:00:00" }	

GET	/Consumer/SystemInformation/LicenseExpiry/Product/{product}	Returns expiration date for given product
		
		Return JSON
		
		{
		"DaysLeft": 20,
		"ExpiryDateTime": "2017/07/27 01:00:00"
		}
			None

GET	/Consumer/SystemInformation/IsConsumerLicensed/Name/{name}	**Minimum API version 5.0** Returns a flag indicating if given consumer is licensed.	None
GET	/Consumer/SystemInformation/IsConsumerLicensed/Id/{id}	**Minimum API version 5.0** Returns a flag indicating if given consumer is licensed.	None
GET	/Consumer/SystemInformation/LicensedConsumers	**Minimum API version 5.0** Returns a list of licensed consumers.	None
POST	/Consumer/SystemInformation/License/Reactivate	Forces license reactivation / update cycle	None
GET	/Consumer/SystemInformation/Telemetry	**Minimum API version 5.1** Returns a flat list containing telemetry data	Read permission on Infrastructure
GET	/Consumer/SystemInformation/Telemetry/Csv	**Minimum API version 5.1**	Read permission on Infrastructure

Verb	Request	Notes	Permissions required
		Returns a flat list containing telemetry data, formatted as CSV	

2.3.52 SystemStatistics

Entity - System Statistics - Interrogates the instrumentation statistics of the system

Verb	Request	Notes	Permissions required

GET	/Consumer/SystemStatistics/HighLevel	Returns high level system statistics.
		
		Example Response
		
		{
		ConnectedDeviceCount,
		ConnectedDeviceCountLastSevenDays,
		InstructionInProgressCount,
		InstructionsPendingApproval,
		DeviceAuthorizationRequests,
		License
		{
		DaysLeft,
		ExpiryDateTime
		}
		}
		
		New for API Version 8.0 : The statistics only refer to the devices and instructions that are visible to the user after imposing the Management Group restrictions. This means that the returned *ConnectedDeviceCount* and *ConnectedDeviceCountLastSevenDays* are counting the number of connected devices *that are visible to the user*, and similarly *InstructionInProgressCount*, *InstructionsPendingApproval* and *ScheduledInstructionsPendingApproval*
			None required

Verb	Request	Notes	Permissions required
		refer to the instructions that are visible to the user.	

GET	/Consumer/SystemStatistics/Detail	Returns detailed system statistics.
		Example Response
		{
		ConnectedDeviceCount,
		AllDeviceCount,
		ConnectedDeviceLastSevenDaysCount,
		QuestionInProgressCount,
		AllQuestionCount,
		QuestionResponseAvailableCount,
		ActionInProgressCount,
		AllActionCount,
		ActionResponseAvailableCount,
		AllEventCount,
		EventsInProgressCount,
		EventsResponseAvailableCount
		}
		Obsolete
		The eventing infrastructure was deprecated in v3.0 and fully removed in v4.2. This means that the two fields **EventsInProgressCount** and **EventsResponseAvailableCount** will not be available from this version onwards.
		New for API version 8.0 : The statistics now only refer to the devices and instructions that are visible to the user after imposing the Management Group restrictions.
			None required

POST	/Consumer/SystemStatistics/ProjectedInstructionStatistics	Returns projected statistics for a given instruction definition.	Execute permission

Verb	Request	Notes	Permissions required

		Example Request
		{
		Id,
		Expression
		}
		Where Id is the Id of the instruction definition and Expression can be used as an option to pass in a scope expression.
		Example Response
		{
		Id
		TotalRuns
		TotalDevices
		EstimatedCount
		EstimatedBytesReceived
		EstimatedAvgExecTime
		EstimatedBytesSent
		EstimatedRowInserts
		EstimatedRowProcessed
		EstimatedSuccessRespondents
		EstimatedSuccessNoDataRespondents
		EstimatedErrorRespondents
		EstimatedNotImplementedRespondents
		EstimatedResponseTooLargeRespondents
		InstructionId
		}
		New for API version 8.0 : When computing the projected statistics, the API now only takes into account the existing statistics that were captured from instructions that are accessible to the user in accordance to the Management Groups.
			on the product pack the instruction definition came from

2.3.53Tasks

Entity - Tasks- handles Tasks and Task Groups

Verb	Request	Notes	Permissions required
GET	/Consumer/Tasks/GetAll	**NO LONGER AVAILABLE IN 3.1** Returns all tasks within the system. Will return a collection of root-level task groups recursively populated with children.	
GET	/Consumer/Tasks/ByGroupId/{id}	**NO LONGER AVAILABLE IN 3.1** Returns a single Task Group, recursively populated with children.	
GET	/Consumer/Tasks/Parent/{id}	Returns all Task Groups and Tasks whose Parent is {id}. It is NOT populated recursively with children. Empty Task Groups (meaning no directly contained Tasks and no child Task Groups) are NOT returned. If **Id** is omitted, the API returns the top-level Task Groups.	The only Tasks returned are those where the calling user has permissions to view the Instruction associated with the tasks.

2.3.54 TriggerTemplates

Entity - Trigger templates

Minimum API version 4.0

Vert	Request	Notes	Permissions required
GET	/Consumer/TriggerTemplates/{id}	Returns a specific Trigger template	Requires 'Read' permission on 'GuaranteedState' securable type
GET	/Consumer/TriggerTemplates	Returns all Trigger templates	Requires 'Read' permission on 'GuaranteedState' securable type

Vert	Request	Notes	Permissions required
POST	/Consumer/TriggerTemplates/Search	Returns Trigger templates that match the search parameters specified • filter - filter expression.To learn about how to define filter check Using scope and filter expressions page. • sort - sort expression. To learn about how to define sort criteria check Sort Definition page. • start - starting index to support pagination. It begins from 1. • pagesize - number of results to fetch Allowed filter columns: • Name • Type • ReadablePayload • Category Allowed sort columns: • Name • Type • Category	Requires 'Read' permission on 'GuaranteedState' securable type
POST	/Consumer/TriggerTemplates	Creates a Trigger template	Requires 'Write' permission on

Vert	Request	Notes	Permissions required
			'GuaranteedState' securable type
PUT	/Consumer/TriggerTemplates	Updates a Trigger template	Requires 'Write' permission on 'GuaranteedState' securable type
DELETE	/Consumer/TriggerTemplates/{id}	Deletes a Trigger template	Requires 'Delete' permission on 'GuaranteedState' securable type

2.4 Using scope and filter expressions

How to use create Expression objects used by many API endpoints as filter or scope definitions.

Many API endpoints accept an expression that defines a search filter or scope for an instruction.

To use such API endpoint you have to create a JSON object describing that expression and send it in the body or your request. Because these expressions can be very complex, they are not used in query strings of GET requests and only utilized in the body of a POST or PUT request.

Expressions used both for filtering and scope definition have the same structure, so this page applies to both types. Attributes of filter expressions depend on the API endpoint (and the entity bring filtered) and will be listed in the endpoint's documentation. Scope attributes are pre-defined and more information can be found here: Defining the scope.

Lastly, many API endpoints which allow filtering also allow sorting. Sort expressions have a different syntax and more information on that can be found on theSort Definitionpage.

2.4.1 Building an expression

2.4.1.1 Anatomy of an expression

Expressions can be built using two types of elements: an expression element (sometimes called a 'leaf') and an expression tree object (called a 'node'):

- An expression element (leaf) - used to define a single condition on an attribute. Elements cannot have any child elements
- An expression tree object - containing a list of expression elements with a logic operator that's applied to that list (called a 'node'). Please note that the operator is applied to all elements on the list.

You should be aware that:

- The expression tree object does not deal with attributes - instead it performs a logic operation on a collection of elements and / or other expression tree objects

- If your expression has only one element in it (for example, just one attribute to filter or scope on), you should send a single expression element without the encompassing expression tree object (see below)
- You can nest expressions to create a multi level tree, an expression tree object can have elements and other tree objects as operands.

2.4.1.2 Available expression operators

Available operators depend on whether you're dealing with an expression element or expression tree object.

Expression element (leaf)

The available operators are essentially SQL operators.

You can use following the operators:

Operator
Like
== (equality)
!= (inequality)
>
<
>=
<=

> **Note**
>
> Not all operators work with all data types.

Expression tree object (node)

The only available logical operators are AND, OR and NOT.

> **Note**
>
> NOT is a unary operator and can only have one operand.

2.4.2 Expression object

Expression element

```
{
        "Attribute": "",
```

```
        "Operator": "",
        "Value": "",
        "DataType": ""
}
```

- Attribute - the columns or element to scope or filter on
- Operator - defines the operator for the expression (from the above list)
- Value - the expression
- DataType - defines the attribute's data type.

The DataType property can be omitted in data sent to the API. It's designed for use by client code to validate the value set for given element.

Supported data types

- string
- integer
- datetime
- double
- boolean

Handling boolean values

Boolean values can only be used with the equality operator comparing against words 'false' and 'true'.

2.4.2.1 Expression tree object

Expression tree

```
{
        "Operator": "",
        "Operands": []
}
```

Operator - defines the logical operator applied to the operands, it can be AND, OR and NOT

Expression - tree object follows normal logic operator rules, for example a tree object with an 'AND' will only return true if all expressions inside it are true while an object with 'OR' will return true if at least one expressions inside it is true.

'NOT' operator is a unary operator which negates whatever its sole operand evaluates to.

Operands - is an array of Expression elements defined above.

2.4.3 Examples

2.4.3.1 Single element expression

An expression that limits a string element called OsType to a value of Windows.

Json for a single element expression

```
{
        "Attribute": "OsType",
        "Operator": "==",
        "Value": "Windows"
}
```

2.4.3.2 Multi element expression

An expression tree consisting of three expression objects.

- The first limits a string element called OsType to a value of Windows
- The second limits an integer element called OsVer to a value of 7
- The third one limits a string element called DeviceType to a value of Desktop.

Json for a multi element expression

```
{
        "Operator": "AND",
        "Operands": [
        {
                "Attribute": "OsType",
                "Operator": "==",
                "Value": "Windows"
        },
        {
                "Attribute": "OsVer",
                "Operator": "==",
                "Value": "7"
        },
        {
                "Attribute": "DeviceType",
                "Operator": "==",
                "Value": "Desktop"
        }
        ]
}
```

2.4.3.3 Multi level expression

An expression tree with multiple levels.

This expression can be used to find computers running Windows OS and are either a desktop or laptop.

The outer level of this tree has two elements - the first limits the OsType to Windows, the other contains the inner expression, which itself limits DeviceType to either desktop or laptop.

JSON for multi level expression

```
{
    "Operator": "AND",
    "Operands": [
    {
        "Attribute": "OsType",
        "Operator": "==",
        "Value": "Windows"
    },
    {
        "Operator": "OR",
        "Operands": [{
            "Attribute": "DeviceType",
            "Operator": "==",
            "Value": "Laptop"
        },
        {
            "Attribute": "DeviceType",
            "Operator": "==",
            "Value": "Desktop"
        }]
    }]
}
```

2.4.3.4 Negating an expression

Negating an expression.

To get only devices which are not Windows desktops you can write an expression selecting Windows desktops and then negate it.

```
{
    "Operator": "NOT",
    "Operands": [{
        "Operator": "AND",
        "Operands": [{
            "Attribute": "OsType",
            "Operator": "==",
            "Value": "Windows"
        },
        {
            "Attribute": "DeviceType",
            "Operator": "==",
            "Value": "Desktop"
        }]
    }]
}
```

2.4.4 Defining a filter

In order to use scope you have to define an expression as explained in[using scope and filter expressions](#).

2.4.4.1 Where are expressions used

There are three main use cases for expressions.

They all share the same syntax as seen on [Using scope and filter expressions](#) page.

In each case, data type field should be null when the expression in sent to the API. Any value passed in that field will be ignored.

2.4.4.1.1 Expressions used to define API filters

Many API endpoints support filtering and sorting of data they return. These often (but not always) include the word 'Search' in their URI, like '/Consumers/Search' (see [Consumers](#) page).

In this case, the attribute in the expression should be one of the columns given endpoint allows filtering on. Documentation for each API endpoint will list which columns are supported in a filter and sort expressions.

2.4.4.1.2 Expressions used to define scope

Scope expressions have a pre-defined set of columns that can be used as attributes. Please see the [Defining the scope](#) page.

Scope expressions are used by endpoints like '/Consumer/Devices/Scope'.

2.4.4.1.3 Expressions used to define instruction result filters

When an instruction is issued it can have a filter applied to its results to better refine the result set. Likewise, a follow up instruction can filter results of the previous instruction to make sure the follow up is executed only when necessary.

[Results filter for an instruction](#) page discusses this use case in greater details.

2.4.4.2 Results filter for an instruction

When issuing an instruction you can define a filter that's applied to the results. When issuing a follow up instruction, you can define a filter to apply to the results of the previous instruction in the follow up chain.

2.4.4.2.1 Use case for filters

Results filters allow you to create a filter that will be evaluated by the client before sending the data back to Tachyon server.

This help cut the "noise" that an instruction may generate by removing responses that you know you are not interested in. They are best used in a situation where you are interested in a subset of data that the instruction would normally returned and that subset can be identified (and the filter specificed) before the instruction is issued. So this feature is useful when you know that you are only interested in a subset of information returned and you want to the agents not to send data you know is irrelevant to what you're looking for.

Filters can be applied to any instruction that returns data and work on the data returned by the instruction.

If a response produced by an agent is filtered down to 0 rows (so there is no data the matches the filter) the agent will return "Success no content". This means that Tachyon server will only get filtered data so this feature should not be used when you want to examine the data set using different filters. In that situation it is best to issue an instruction without the filter and apply filtering when retrieving responses, or you can issue multiple copies of the instruction, each with a different results filter.

2.4.4.2.1.1 Example use case

Let's say you want to check if any of the devices in your estate have a specific drived with an identified vulnerability installed.

You could simply issue the "1E-Explorer-TachyonCore-DeviceDrivers" instruction, but that would return all drivers and in this instance you know exactly what driver you're insterested in.

Results filters allow you to issue an instruction which will returns the rows you want.

2.4.4.2.2 How filters work

2.4.4.2.2.1 Results filter

A "results filter" is executed at the end of an instruction and filters the results much like a SQL "WHERE" clause would.

This filter can use any column that exist in the instruction's schema.

The instruction is executes as normal and produced a return data set. Then the agent applies the filter to the dataset and sends back the matching rows. If no matching rows were found, it sends back "Success no content".

2.4.4.2.2.1.1 Example

The flow here is simple:

- Execute the instruction are prepare return dataset
- Apply filter to the return dataset
- Return filtered dataset

2.4.4.2.2.2 Previous results filter

A "previous results filter" can only be used in a follow-up scenario. It work on the result set returned by the **previous** instruction, so the instruction that you are issuing a follow up to.

As a quick refresher, a follow-up instruction forms a chain of two or more instructions. When a follow-up instruction is executes, each element in chain is executed if the previous one returned data, so if its final select statement produced a dataset that wasn't empty. This means that only the dataset from the last link in the chain, so the follow-up instruction itself is returned. Dataset from the internediate instructions in the chain are there for the agent to make a decision if it should continue executing instruction in the chain and are **not** returned to the Tachyon server.

This means that this filter uses the columns from the schema of the **previous** instruction and **not** the one you are issuing.

This means that "previous results filter" is executed at the end of the **previous** instruction and **before** the follow up instruction. It filters the results of the previous instruction much like a SQL "WHERE" clause would.

If there is any data left in the dataset after it was filtered, the follow up instruction is executed.

2.4.4.2.2.2.1 Simple example

Let's assume you have issed an instruction called "Search-For-Stuff" and now you are issuing a follow-up to that called "Fix-Stuff", but you want to fix only certain things identified by "Search-For-Stuff" instruction, so when issuing "Fix-Stuff" you apply a previous results filter to it. The flow here would be:

- Execute "Search-For-Stuff" instruction and prepare its return dataset
- Apply the Previous Results Filter to the return dataset
- If filtered dataset has no rows, return "Success no content" immediatelly. If it does have rows, continue the execution
- Execute "Fix-Stuff" instruction and prepare its dataset
- Return the dataset from the "Fix-Stuff" instruction

2.4.4.2.2.2.2 Complex example

Let's assume you have a slightly more complex chain of events. You issue an instruction called "Asset-Inventory", then you issue "Software-Inventory" instruction as a follow up to "Asset-Inventory" and lastly you issue "Update-Software" instruction as a follow up to "Software-Inventory" but you specify a previous results filter so that only specific software from a specific manufacturer gets updated. The flow here would be:

- Execute "Asset-Inventory" instruction and prepare its return dataset
- If the return dataset has no rows, return "Success no content" immediatelly. If it does have rows, continue the execution
- Execute "Software-Inventory" instruction and prepare its return dataset
- Apply the Previous Results Filter to the return dataset of "Software-Inventory" instruction.
- If filtered dataset has no rows, return "Success no content" immediatelly. If it does have rows, continue the execution
- Execute "Update-Software" instruction and prepare its dataset
- Return the dataset from the "Update-Software" instruction

2.4.4.2.2.3 Combining filter types

Both filter types can be combined and they execute in a specific sequence:

- 1st instruction executes producing Dataset A
- Results filter of the 1st instruction executes on Dataset A producting Dataset B

- Previous results filter of the 2nd instruction executes on the Dataset B producing dataset C
- 2nd instruction executes producing Dataset D
- Results filter of the 2nd instruction executes on the Dataset D producing Dataset E
- Dataset E is returned

Of course, if any of the datasets A through D were empty, a "success no content" would be returned immediately.

As you can see each filter, be that Results filter or Previous results filter, is executed against the last dataset created. This is why 2nd instruction's Previous Results Filter is executed against the dataset that was the result of he 1st instruction's Results Filter.

If 1st instruction had no Results Filter defined, 2nd instruction's Previous results filter would still execute against the latest dataset - the dataset returned by the 1st instruction (in the example above that's Dataset A).

2.4.4.2.3 Using results filter

Results filter will be applied to any data that would normally be returned after the instruction was run. Because of that, it can only use the columns returned by the instruction, so columns from instruction's schema.

You can define a filter on any and all of those columns, but only on those columns. If you define a filter, only results matching the filter will be returned by the client. If a client evaluates a filter and discovers it has no data that matches it, it returns "Success no content" message.

This filter is useful when the instruction you want to issue does not have a parameter that would allow you to narrow down the results.

2.4.4.2.3.1 Example

For example, for an instruction returning general information about each computer and normally, when unfiltered, gives the following results:

Caption	Free Physical Memory	Locale	Number of users	Registered User
Server1	8192	en-us	7	Administrator
Server2	8192	en-us	4	Administrator
Desktop1	1024	en-us	1	JohnDoe
Desktop2	8192	en-us	1	JaneDoe

The instruction itself does not allow any parameters, but to see computers for example, with less than 2GB of RAM the results can be defined with this filter:

Filter

```
{
    "Attribute": "Free Physical Memory",
    "Operator": "<=",
    "Value": "2048"
}
```

This filters the results to show computers with less than 2GB of RAM:

Caption	Free Physical Memory	Locale	Number of users	Registered User
Desktop1	1024	en-us	1	JohnDoe

2.4.4.2.4 Using previous results filter

The previous results filter is applied to the results of the previous instruction in the follow up chain, and before the actual instruction being issued is run.

This filter is useful if you want to issue a follow up instruction that will be executed only by agents that would send results that fulfill a given condition.

It is worth noting that unlike scope, a filter will not limit the number of agents that receive the instruction, but will limit the results being sent back.

2.4.4.2.4.1 Example

These results are for an instruction that returns network adapters installed on each computer.

The results contain the columns: Description, MAC Address, Manufacturer and Physical Adapter.

Device Name	Description	MAC Address	Manufacturer	Physical Adapter
machine1.local	Intel(R) 82579LM Gigabit Network Connection	18:03:73:34:25:93	Intel Corporation	true
machine1.local	Hyper-V Virtual Ethernet Adapter	12:12:22:12:11:11	Microsoft	false
machine2.local	Intel(R) 82579LM Gigabit Network Connection	18:03:73:35:22:98	Intel Corporation	true
server1.local	X710-DA4	22:21:95:34:33:53	DELL	true
server2.local	NC1000TX-G	54:56:32:77:34:45	Dynamode	true

To find out what driver version each computer uses for the Intel 82579LM you can issue an instruction that returns the driver version and defines the following filter for the previous results:

Previous Results Filter

```
{
        "Operator": "AND",
        "Operands": [{
                    "Attribute": "Description",
                    "Operator": "like",
                    "Value": "%82579LM%"
                },
                {
                        "Attribute": "Manufacturer",
                        "Operator": "like",
                        "Value": "Intel%",
                }
}
```

The previous results filter only cause the agents that would return a matching result to execute the follow-up instruction. This removes any unwanted results from other NICs.

Device Name	Description	Driver Version
machine1.local	Intel(R) 82579LM Gigabit Network Connection	1.0.0.1
machine2.local	Intel(R) 82579LM Gigabit Network Connection	1.0.0.5

2.4.4.2.5 Putting it all together

You can combine a previous results filter with the results filter on the same instruction.

For example, to make sure network drivers on all computers on your estate are up to date:

1. Issue an instruction to return network adapters.
2. Only return physical adapters.
3. Set the results filter to filter out virtual adapters.

First instruction's results filter

```
{
    "Attribute": "Physical Adapter",
    "Operator": "==",
    "Value": "true",
}
```

This returns a list of results of physical network adapters on all computers:

Device Name	Description	MAC Address	Manufacturer	Physical Adapter
machine1.local	Intel(R) 82579LM Gigabit Network Connection	18:03:73:34:25:93	Intel Corporation	true
machine2.local	Intel(R) 82579LM Gigabit Network Connection	18:03:73:35:22:98	Intel Corporation	true
server1.local	X710-DA4	22:21:95:34:33:53	DELL	true
server2.local	NC1000TX-G	54:56:32:77:34:45	Dynamode	true
server3.local	X710-DA4	23:89:82:34:22:00	DELL	true

On this list there's a number of DELL adapters which have had recent driver updates because of stability issues. Because drivers older than version 11 can be problematic, you'd need see which computers require a driver update.

You issue an instruction returning network driver details.

Because you only want DELL adapters returned, you apply a previous result filter to filter out everything else:

Follow up instruction's previous resoults filter

```
{
        "Operator": "AND",
        "Operands": [{
                    "Attribute": "Description",
                    "Operator": "==",
                    "Value": "X710-DA4"
                },
                {
                        "Attribute": "Manufacturer",
                        "Operator": "==",
                        "Value": "DELL",
                }
}
```

To find versions older than 11, apply a results filter onto the same instruction:

Follow up instruction's results filter

```
{
    "Attribute": "Driver version",
    "Operator": "<",
    "Value": "11",
}
```

It returns these results:

Device Name	Description	Driver Version
server1.local	X710-DA4	9

You could use these results to update the drivers on those specific computers.

2.4.5 Defining the scope

To use scope you must define an expression, read Using scope and filter expressions for more information.

2.4.5.1 Available scope attributes

The system has a number of predefined attributes for use in a scope expression.

Type	Name
Attribute	ostype
	osver
	name
	rammb
	osvertxt
	agentver
	vrplatform
	model
	fqdn
	devtype
	timezone
	lastboot
	tagtxt

Type	Name
managementgroup	
criticality	

2.4.5.1.1 Scope expression limitations pre v3.2

The two scope expression limitations for Tachyon pre v3.2 are the:

- Only operator supported in the expression tree object is AND
- System only supports single level expression trees, for example you cannot have an expression tree object and an operand for another expression tree object.

2.4.5.2 Handling tags

Tags are a specific case. Tags always use the tagtxt attribute name.

To scope on tags leave the Operator field blank (""), like:

Tag element Json

```
{
        "Attribute": "tagtxt",
        "Operator": "",
        "Value": "key=value"
}
```

> **Note**
>
> Tag names are not case-sensitive but tag values are, so you will get an error if you specify a tag value whose case does not match one of the stored values defined for that tag.
>
> Also there cannot be spaces around the = sign, if you specify "key = value" then you will get an error

2.4.5.3 Handling Management Groups

Management groups are another special case. They always use the managementgroup attribute name.

The only operator they support is == and do not have a data type.

The value is the UsableId, not the Name or Id of the Management group.

Tag element Json

```
{
        "Attribute": "managementgroup",
        "Operator": "==",
```

```
        "Value": "40497EF0F9C344BE937E2813D22457E9"
}
```

2.4.5.4 Handling Criticality

When criticality is used in a scope expression the value needs to be the numerical value, not the "friendly" name seen in the UI.

Mapping between the numbers and names can be found in the database in the GlobalSettings table, row with the name "CriticalityMapping".

The default mapping can be seen below.

> **Note**
>
> Double check these mappings as they might have been changed in your installation of Tachyon.

Numerical Value	Friendly name
0	Undefined
1	Non-critical
2	Low
3	Medium
4	High
5	Critical

2.4.6 Sort Definition

Many API endpoints support filtering and sorting of the data they return.

These endpoints usually, but not always, have a 'Search' word in their URI, like '/Consumers/Search'.

2.4.6.1 Building sort expression

Sort expressions are defined as JSON. It's defined as a collection of sort specs, for example:

Sort Expression

```
[
  {
     "Column":"Column1",
     "Direction":"ASC"
  },
  {
```

```
    "Column":"Column2",
    "Direction":"DESC"
  }
]
```

- ASC - sort in ascending order - the default if no direction is specified
- DESC - sort in descending order.

2.5 How to guides

Section describing some basic and common Tachyon operations using the Consumer API and Consumer SDK. It's aimed at developers who want to write code that integrates with Tachyon. It focuses on what APIs to use and how to use them so you can carry out common operations. This section does not cover details about how Tachyon internal operations work.

Instead, Tachyon is treated as a black box, which developers can use without detailed knowledge of how it works. For full details about Tachyon read more about how Tachyon works in the Tachyon Platform documentation Tachyon Platform 8.1.

To get the most from this section you should understand web based APIs, their associated terminology and concepts.

2.5.1 Authenticate against the 1E Platform and obtain a token

This page describes the interactive authentication and code samples that can be adapted by the customer while writing their own integration software.

It should also be understood that these pages will use {apiRoot} symbol for the Root of the Portal (not Consumer) API, as it is Portal API that deals with authentication requests. The root of that API will look like https://your.tachyon.server/Tachyon/api (for 9.X and 23.X) or https://your.tachyon.server/api for versions 24.1 and later. (note the word “Tachyon” is missing in 24.1 and later versions). It is assumed that the code contained on these pages will be adapted by configuring the {apiRoot} correctly and that any other configuration (like JWT certificates and user mapping) have also been carried out.

2.5.1.1 Authentication via IdP’s web page

Interactive authentication is performed via the IdP’s web page and requires username/password and few other steps to complete the authentication. This mode of authentication is the best option for the applications having a UI that allows the user to navigate to IdP’s page and where explicit authentication is not an issue.

2.5.1.2 Authentication using a JWT

Non-interactive authentication using JWT is performed in-code and does not require any usernames or passwords. This authentication mode can be used for applications where UI cannot display IdP’s login page or where such display would be an issue. It can also be used for service accounts or to group all (or some) users of an application behind a single (or several) accounts in the 1E Platform without having a one-to-one mapping between an IdP user and a 1E Platform principal.

2.5.1.3 Implementing interactive authentication

This page describes how to perform interactive authentication against the 1E Platform and your IdP in order to obtain a token.

2.5.1.3.1 How it works

Authentication process requires a secret (as explained in the later steps) to be created and has the following steps:

- Initial request for authentication is made to the 1E Platform that returns **state** and a **URL** that needs to be called in order to perform authentication with the IdP
- Visit the **URL** provided in the previous step and perform the required authentication
- Poll for status change to establish in the background when the token is ready, or the process is failed
- Upon successful authentication, retrieve the token

2.5.1.3.2 Before we begin

Authentication process requires a secret called a **nonce**. This is usually an array of random 32 bytes that should be created before the authentication process is started.

Once you have the array, you need to create two things:

1. A URL-safe Base64 encoded of the nonce. Let's call this BASEDNONCE
2. A URL-safe Base64 encoded SHA256 of the nonce (you need to obtain SHA256 of the nonce and then Base64 encode it). Let's call this HASHEDNONCE

The original nonce is not used afterwards and only those two values, BASEDNONCE and HASHEDNONCE are used.

It is strongly advised that nonce is randomized each time a new request is made, as reusing a nonce decreases security of the authentication process.

2.5.1.3.3 Authentication request

Initial request should be made to **{apiRoot}/Authentication/RequestAuthentication** and this request should have a query string parameter called **ecpn**. You should put the HASHEDNONCE as the value of the parameter.

This means that the initial request will look something like this:

```
https://your.tachyon.server/api/Authentication/RequestAuthentication?ecpn=e0bebd228199934258l4866b62701e2919ea26f1370499c1037b53b9d49c2c8a
```

and you will receive something like this from the API as a response:

```
{
    "AuthenticationUrl": "https://login.microsoftonline.com/06c15aaf-bd3b-4eac-aac9-ef6e9ce805b0/oauth2/v2.0/authorize?response_type=code&prompt=login&client_id=9858ac43-555e-4500-9ccd-bac33e1ddfba&state=5PVOGbg0lh_Juj5P9D7dLAAhkY4N-f4Kg
```

XJ8jgH88dQ.ZDQ2ZDQyNDNjMTM3ZGZhY2YzNzNmYzBhOGFmYmYzNTNiZDJjNzU4OGVjNTQ0YTY4NTdj
YjExOTljNjFhMGE1Ng.in_yI295_1yUC-nvqdHgIS_-3gOVVj9G4SANupo8JCdWoqaHb7iqYsPEJvqE
Ka0EzrualgWociVuL63HttqbN79bgy_ZRHXO3-EYBwn_JE6dvIckpOBkAKQghdMyCn4ctidAhnkIF0I
ER8pHI4aNuMVRa4_um7KILM4GwW36luVOEgP_3xpGrofgKp_dDHYFuJq6a216qok65_vGK_fzc6sES9
pcau9axE-1DWZ2a_JTwXZy0K3xx38hx4oxSfb5cnFn-e1A0hP06In4dWImF6t_6Lawc06-Ya84vPUmg
2SR45NwudUJamHdjsC_G_rqWIWi98aIOEqjJGJDZvOGNg&scope=openid&redirect_uri=https:/
/your.tachyon.server:443/Portal/api/Authentication/IdentityProviderRedirect&cod
e_challenge=h1oqLd3TmMmk82j_O72FIZWN_4vKFN9mEDXndQpLAdw&code_challenge_method=S
256",
    "State": "5PVOGbg0lh_Juj5P9D7dLAAhkY4N-f4KgXJ8jgH88dQ.ZDQ2ZDQyNDNjMTM3ZGZhY
2YzNzNmYzBhOGFmYmYzNTNiZDJjNzU4OGVjNTQ0YTY4NTdjYjExOTljNjFhMGE1Ng.in_yI295_1yUC
-nvqdHgIS_-3gOVVj9G4SANupo8JCdWoqaHb7iqYsPEJvqEKa0EzrualgWociVuL63HttqbN79bgy_Z
RHXO3-EYBwn_JE6dvIckpOBkAKQghdMyCn4ctidAhnkIF0IER8pHI4aNuMVRa4_um7KILM4GwW36luV
OEgP_3xpGrofgKp_dDHYFuJq6a216qok65_vGK_fzc6sES9pcau9axE-1DWZ2a_JTwXZy0K3xx38hx4
oxSfb5cnFn-e1A0hP06In4dWImF6t_6Lawc06-Ya84vPUmg2SR45NwudUJamHdjsC_G_rqWIWi98aIO
EqjJGJDZvOGNg"
}

2.5.1.3.4 Examining the response to initial authentication request

The response you received contains two properties.

- First is the authentication URL. You should take this URL and using your web browser, navigate to display the IdP login page.
- Second property is called State. This identifies your request and you will need this to check what is happening with your request and to retrieve the token.

2.5.1.3.5 Authenticating against the IdP

Authentication against your IdP has to be done outside the bounds of the 1E Platform, as the IdP is an independent system. You can handle it however you like, but the process must be completed successfully in order for the authentication process to complete. As the IdP will call 1E Platform when you have successfully authenticated. If that call never comes, the token will not be issued.

2.5.1.3.6 Checking the status of your authentication request

After the authentication process has been initiated, you should start polling to check if it has completed.

You do that by calling GET on **{apiRoot}/Authentication/CheckAuthenticationState** while providing a payload with the state you have received in the response to the authentication request:

{
    "State" : "5PVOGbg0lh_Juj5P9D7dLAAhkY4N-f4KgXJ8jgH88dQ.ZDQ2ZDQyNDNjMTM3ZGZh
Y2YzNzNmYzBhOGFmYmYzNTNiZDJjNzU4OGVjNTQ0YTY4NTdjYjExOTljNjFhMGE1Ng.in_yI295_1yU
C-nvqdHgIS_-3gOVVj9G4SANupo8JCdWoqaHb7iqYsPEJvqEKa0EzrualgWociVuL63HttqbN79bgy_
ZRHXO3-EYBwn_JE6dvIckpOBkAKQghdMyCn4ctidAhnkIF0IER8pHI4aNuMVRa4_um7KILM4GwW36lu
VOEgP_3xpGrofgKp_dDHYFuJq6a216qok65_vGK_fzc6sES9pcau9axE-1DWZ2a_JTwXZy0K3xx38hx
4oxSfb5cnFn-e1A0hP06In4dWImF6t_6Lawc06-Ya84vPUmg2SR45NwudUJamHdjsC_G_rqWIWi98aI
OEqjJGJDZvOGNg"
}

Warning

Do not provide any other data with these requests. Specifically, do not provide BASEDNONCE as seen in the next paragraph. You can only attempt token retrieval once. If you don't do it correctly you will not be able to retrieve the token again and you will have to start the authentication process again.

The API will respond with following data:

```
{
  "Status":"AuthenticationRequested",
  "Data":""
}
```

The status can be one of following values:

- AuthenticationRequested - means the authentication is still in progress but it hasn't finished yet. Token isn't ready and you should keep pooling
- AuthenticationResultNotAvailable - this means this authentication attempt has failed and you will not be able to retrieve the token. You have to start a new authentication process. The system does not return a specific reason on purpose as a security measure
- AuthenticationSuccessful - means the authentication has been successful and the token can be retrieved

2.5.1.3.7 Retrieving the token

Token is retrieved via the same endpoint you used to check the authentication state, but you have to also provide BASEDNONCE:

```
{
    "State" : "5PVOGbg0lh_Juj5P9D7dLAAhkY4N-f4KgXJ8jgH88dQ.ZDQ2ZDQyNDNjMTM3ZGZh
Y2YzNzNmYzBhOGFmYmYzNTNiZDJjNzU4OGVjNTQ0YTY4NTdjYjExOTljNjFhMGE1Ng.in_yI295_1yU
C-nvqdHgIS_-3gOVVj9G4SANupo8JCdWoqaHb7iqYsPEJvqEKa0EzrualgWociVuL63HttqbN79bgy_
ZRHXO3-EYBwn_JE6dvIckpOBkAKQghdMyCn4ctidAhnkIF0IER8pHI4aNuMVRa4_um7KILM4GwW36lu
VOEgP_3xpGrofgKp_dDHYFuJq6a216qok65_vGK_fzc6sES9pcau9axE-1DWZ2a_JTwXZy0K3xx38hx
4oxSfb5cnFn-e1A0hP06In4dWImF6t_6Lawc06-Ya84vPUmg2SR45NwudUJamHdjsC_G_rqWIWi98aI
OEqjJGJDZvOGNg"
    "Nonce": "your BASEDNONCE goes here"
}
```

If the data you have provided is correct you will receive the token:

```
{
  "Status":"AuthenticationSuccessful",
  "Data":"your token will be here"
}
```

Warning

You have only one attempt to retrieve the token. This means that for given state (for given request) the system will allow you to make precisely ONE call to

{apiRoot}/Authentication/CheckAuthenticationState where your provide both State and Nonce. If you provide wrong data or make the call too early, you will not be able to retrieve the token and you will have to start a new authentication request.

2.5.1.3.8 What if something goes wrong?

Every API call response should be examined for errors. In most cases you will receive standard 1E Platform errors (see Handling errors returned by Tachyon for more details), but there might be situations where something else is returned.

2.5.1.3.9 Consumer SDK support

Currently Consumer SDK does not support authentication and you have to write you own code. We're providing examples to help you with that.

2.5.1.3.10 Implementing interactive authentication - C# example

This page contains a code example written in C# that demonstrates interactive authentication. This code sample should be adapted by the customer when writing their own integration software.

Note

Any code given is without any warranty and is meant only for demonstration and is not production ready code. This means that the code has brevity omitted things like error and exception handling.

2.5.1.3.10.1 Expand for C# example

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;

namespace InteractiveAuthenticationClient
{
    public class Program
    {
        private const string ApiRoot = "https://your.tachyon.server.address/api
/";
        private const string RequestAuthenticationEndpoint = "Authentication/Re
questAuthentication?ecpn={0}";
        private const string CheckAuthenticationStateEndpoint = "Authentication
/CheckAuthenticationState";

        static void Main(string[] args)
        {
            Console.WriteLine("Creating authentication nonce...");
            // Prepare nonce
            var nonce = CreateNonce(); // original nonce, never supplied to any
```

```
api
            var basedNonce = ToUrlSafeBase64(nonce); // base64 encoded nonce, u
sed when retrieving a token
            var hashedNonce = CreateSha256(nonce); // hash of the original nonc
e, never supplied to any api
            var basedHashedNonce = ToUrlSafeBase64(hashedNonce); // base64 enco
ded has of the nonce, to be supplied in the original authentication request
            using (var client = new HttpClient())
            {
                var endpoint = ApiRoot + string.Format(RequestAuthenticationEnd
point, basedHashedNonce);
                Console.WriteLine($"Calling {endpoint} to initiate authenticati
on");
                AuthRequestResponse obj = Get<AuthRequestResponse>(client, endp
oint);
                if (obj != null)
                {
                    if (!string.IsNullOrEmpty(obj.AuthenticationUrl))
                    {
                        Console.WriteLine($"Received authentication URL {obj.Au
thenticationUrl} and state {obj.State}. Opening web browser for the user to com
plete authentication process.");
                        OpenBrowser(obj.AuthenticationUrl);
                        string token = LoopForToken(obj.State, basedNonce);
                        if (token != null)
                        {
                            Console.WriteLine($"Token retrieved: {token}");
                        }
                        else
                        {
                            Console.WriteLine("Token could not be retrieved. Pl
ease try again.");
                        }
                    }
                }
            }
        }

        private static string LoopForToken(string objState, string basedNonce)
        {
            var continueLoop = true;
            string retVal = null;
            var endpoint = ApiRoot + CheckAuthenticationStateEndpoint;
            using (var client = new HttpClient())
            {
                while (continueLoop)
                {
                    var payload = new AuthenticationCheckRequestPayload
                    {
                        State = objState
```

```
                    };
                    var result = Post<AuthenticationEvent, AuthenticationCheckR
equestPayload>(client, payload, endpoint);
                    if (result != null)
                    {
                        switch (result.Status)
                        {
                            case "AuthenticationResultNotAvailable":
                            {
                                // There's an issue with the request. Either th
e state is incorrect or you tried to retrieve the token incorrectly or it alrea
dy has been redeemed.
                                // You will need to initiate new authentication
request as you cannot continue with this one.
                                Console.WriteLine("Something went wrong during
the authentication process. Please start a new authentication process.");
                                continueLoop = false;
                                break;
                            }
                            case "AuthenticationRequested":
                            {
                                Console.WriteLine("Authentication process is st
ill in progress. Will try again in a few seconds.");
                                Thread.Sleep(2000);
                                // Authentication request is still being proces
sed.
                                break;
                            }
                            case "AuthenticationSuccessful":
                            {
                                Console.WriteLine("Authentication process has f
inished. Retrieving token...");
                                // Authentication has been successful and token
can be retrieved.
                                var payload2 = new AuthenticationCheckRequestPa
yload
                                {
                                    State = objState,
                                    Nonce = basedNonce
                                };
                                var result2 = Post<AuthenticationEvent, Authent
icationCheckRequestPayload>(client, payload2, endpoint);
                                if (result2 != null)
                                {
                                    retVal = result2.Data;
                                }
                                else
                                {
                                    Console.WriteLine("Failed to retrieve token
.");
```

```
                                }

                                continueLoop = false;
                                break;
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine("Call to retrieve authentication requ
est has failed.");
                        continueLoop = false;
                    }
                }
            }

            return retVal;
        }

        private static byte[] CreateSha256(byte[] buffer)
        {
            using (var sha256 = new SHA256CryptoServiceProvider())
            {
                return sha256.ComputeHash(buffer);
            }
        }

        private static byte[] CreateNonce()
        {
            byte[] nonce = new byte[32];
            Random rand = new Random();
            rand.NextBytes(nonce);
            return nonce;
        }

        private static string ToUrlSafeBase64(byte[] arg)
        {
            var Base64PadCharacter = '=';
            var Base64Character62 = '+';
            var Base64Character63 = '/';
            var Base64UrlCharacter62 = '-';
            var Base64UrlCharacter63 = '_';

            var s = Convert.ToBase64String(arg);
            s = s.Split(Base64PadCharacter)[0]; // RemoveAccount any trailing p
adding
            s = s.Replace(Base64Character62, Base64UrlCharacter62); // 62nd cha
r of encoding
            s = s.Replace(Base64Character63, Base64UrlCharacter63); // 63rd cha
r of encoding
```

```
            return s;
        }

        public static void OpenBrowser(string url)
        {
            try
            {
                Process.Start(url);
            }
            catch
            {
                // hack because of this: https://github.com/dotnet/corefx/issue
s/10361
                if (RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
                {
                    url = url.Replace("&", "^&");
                    Process.Start(new ProcessStartInfo("cmd", $"/c start {url}"
) { CreateNoWindow = true });
                }
                else if (RuntimeInformation.IsOSPlatform(OSPlatform.Linux))
                {
                    Process.Start("xdg-open", url);
                }
                else if (RuntimeInformation.IsOSPlatform(OSPlatform.OSX))
                {
                    Process.Start("open", url);
                }
                else
                {
                    throw;
                }
            }
        }

        private static T Get<T>(HttpClient client, string url)
        {
            var result = client.GetAsync(url).Result;
            if (result.IsSuccessStatusCode)
            {
                var data = result.Content.ReadAsStringAsync().Result;
                if (!string.IsNullOrEmpty(data))
                {
                    T obj = JsonSerializer.Deserialize<T>(data);
                    if (obj != null)
                    {
                        return obj;
                    }
                }
            }
```

```
            return default;
        }

        private static TOut Post<TOut, TIn>(HttpClient client, TIn payload, str
ing url)
        {
            var stream = new MemoryStream();
            JsonSerializer.Serialize(stream, payload);
            stream.Seek(0, SeekOrigin.Begin);
            var reader = new StreamReader(stream);
            var serializedPayload = reader.ReadToEnd();
            reader.Close();
            stream.Close();
            var content = new StringContent(serializedPayload, Encoding.UTF8, "
application/json");
            var result = client.PostAsync(url, content).Result;
            if (result.IsSuccessStatusCode)
            {
                var data = result.Content.ReadAsStringAsync().Result;
                if (!string.IsNullOrEmpty(data))
                {
                    TOut obj = JsonSerializer.Deserialize<TOut>(data);
                    if (obj != null)
                    {
                        return obj;
                    }
                }
            }

            return default;
        }

        private static string Post<TIn>(HttpClient client, TIn payload, string
url)
        {
            var stream = new MemoryStream();
            JsonSerializer.Serialize(stream, payload);
            stream.Seek(0, SeekOrigin.Begin);
            var reader = new StreamReader(stream);
            var serializedPayload = reader.ReadToEnd();
            reader.Close();
            stream.Close();
            var content = new StringContent(serializedPayload, Encoding.UTF8, "
application/json");
            var result = client.PostAsync(url, content).Result;
            if (result.IsSuccessStatusCode)
            {
                return result.Content.ReadAsStringAsync().Result;
            }
```

```
            return default;
        }

        private class AuthRequestResponse
        {
            public string AuthenticationUrl { get; set; }
            public string State { get; set; }
        }

        private class AuthenticationCheckRequestPayload
        {
            public string State { get; set; }
            public string? Nonce { get; set; }
        }

        private class AuthenticationEvent
        {
            public string Status { get; set; }

            public string Data { get; set; }
        }
    }
}
```

2.5.1.4 Implementing non-interactive (JWT) authentication

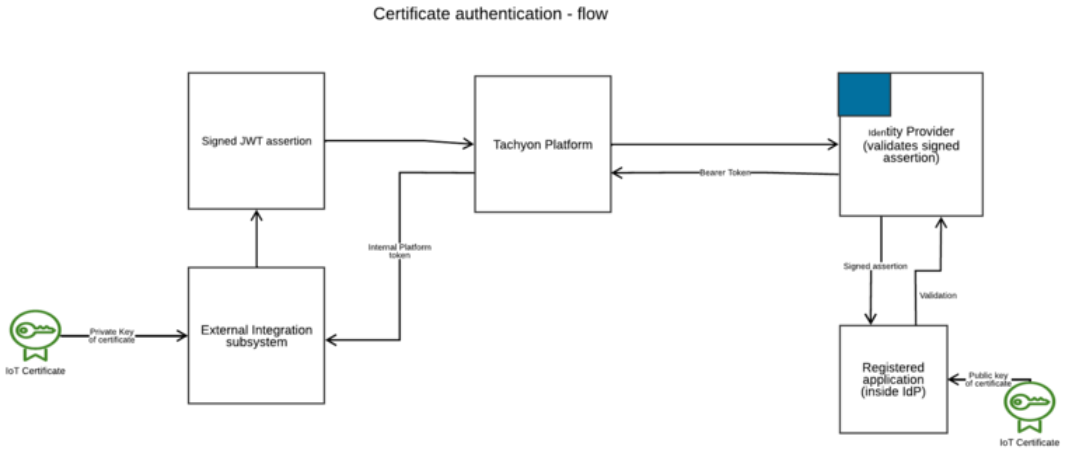
This type of authentication is useful for external applications created by systems integrators that need to interact with various platform components without using interactive browser-based authentication.

2.5.1.4.1 How it works

Authentication is performed based on having access to the private key of a certificate. This can be a self-signed certificate; it does not need to be issued by a trusted Certification Authority. The trust is managed by uploading the public key of the certificate to the Identity Provider you are using. The private key needs to be accessible to the local application that wants to communicate with the Platform.

In the diagram below, you can see at the left-hand side how the private key of the certificate is available to the External Integration subsystem (the application that wants to call the Platform APIs) while at the right-hand side the public key is available to the Identity Provider (abbreviated as IdP in the following text).

In an IdP such as Azure Active Directory, the association of the private key to the IdP is configured by the intermediation of something called a “Registered application” which allows for configuration of an item at the IdP where you assign permissions for external applications.



When the External integration subsystem wants to connect to the Platform, the flow of information is as follows:

1. The External integration subsystem generates a JWT. This is a string of text that encodes the identity of the user and is cryptographically signed. The signature is done with the private key of the certificate.
2. The JWT is sent to an endpoint in the Platform.
3. The platform verifies the JWT locally and only when it is satisfied with local verification it sends the JWT to the IdP, which is able to verify the signature thanks to the public key of the certificate.
4. If the IdP is able to verify that the signature is correct, it notifies the Platform.
5. In the next step, the Platform generates a token and returns it to the integration subsystem.

The Tachyon Token has a limited lifetime. If it is used after its expiration, the API will return a 401 (Access Denied) status to the caller. Contrary to an interactive login, the tokens acquired through the non-interactive process cannot be refreshed. This means that the caller will need to repeat all the previous steps to acquire a new Tachyon Token and then retry the operation using the new token.

Because you are using essentially a certificate to authenticate, the system cannot automatically determine who is authenticating, only that the have a valid “key” (certificate) and supplied correct data.

In order to determine which user is being authenticated, each certificate is mapped to either a specific user or a number of users (via explicit mappings or a wildcard-style mapping). This means that if you plan on authenticating multiple users via JWTs, you have the flexibility of using either one or multiple certificates.

2.5.1.4.2 Configuration

Prior to using this feature, your Platform installation must be configured correctly in order for JWT authentication to work.

While creation of a certificate and installing it inside your IdP is outside the bounds of this article and will depend somewhat on your IdP, as steps for Azure will be different to those for OKTA, we’d like to outline general modus operandi.

You will need a certificate that you can use with your IdP, be that a self-signed certificate or one obtained from a trusted source. As mentioned above, if you are planning on authenticating multiple users, please consider whether you wish to use multiple certificates or not.

Any certificates you wish to use need to be installed in your IdP and the application you wish to use for authentication has to have access to those certificates in order for the IdP to be able to validate incoming requests.

Your organisation may have already pre-configured an authentication application which they wish to use for certificate authentication. In this case, you will access the application whose Client Id has been specified, and ensure that certificates have been associated with that application.

Alternatively you may wish to create a new authentication application that is to be used for exclusively certificate authentication.

In either case, you must ensure that the Platform's tenant configuration has the 'clientassertionappid' member set to the client Id of the authentication application you wish to use and that any certificates you want to use are associated with the application and that the application has access to those certificates.

When you authenticate using a certificate, you will be mapped to a Principal. That Principal's permissions will then control your rights on the Tachyon Platform.

Certificate mapping is set up using the Tachyon PowerShell toolkit to add mapping entries. Each mapping entry associates a certificate with a desired principal.

You can also set up 'wildcard' mappings, which allow a certificate to map to any Principal. This is a privileged operation and should be used with caution, as it allows the certificate owner to operate as any platform Principal.

Alternatively you can map the same certificate multiple times, each time to a different Principal. This allows the certificate owner to request access as any of the mapped Principals. This can be useful if you want the same certificate to be useable in different user contexts.

The process is explained in detail here ADD LINK

2.5.1.4.3 Authentication steps

A JWT has a header and a body. The header and body are encoded in base 64 separately and combined using a period (.) to create a representation suitable for placing in a request header.When signed, the signature (which signs the header plus the body as a whole) is appended as a third chunk of base 64, again separated by a period. This means that a proper JWT request has three parts separated by '.' characters like so: header.body.signature.

Internally, these pieces are encoded as a JSON object. If you were to generate them as classes in C# they would look like this:

```
header = new JwtHeader()
             {
                 alg = "RS256",
                 typ = "JWT",
                 x5t = thumbprint,
                 kid = thumbprint
             };
body = new JwtBody()
             {
                 aud = clientAssertionAudience,
```

```
                exp = ((DateTimeOffset)DateTime.UtcNow + TimeSpan.FromSeconds
(3600)).ToUnixTimeSeconds().ToString(),
                iss = clientAssertionIssuer,
                sub = clientAssertionSubject,
                jti = Guid.NewGuid().ToString()
            };
```

The header members x5t and kid are typically set to the base64-encoded thumbprint of the certificate used to sign the JWT. This allows the receiving identity provider to know which certificate to use when validating the JWT. The Platform will accept any arbitrary value for the kid without requiring it to match the thumbprint of the certificate. This “kid” value is the one you will map to a Principal in Tachyon using the KidToUpnMapping table.

The body members are as follows:

- **aud** - Audience - this is usually set to the URL of the IdP's token issuing endpoint.
- **exp** - Expiry - expiration timestamp for the JWT, defined in Unix Epoch seconds
- **iss** - Issuer - this is usually set to the Application Id in the IdP that will handle the client assertion grant flow and validate the JWT.
- **sub** - Subject - The subject is usually set to be the same as the issuer
- **jti** - This is just a random GUID to make each JWT unique; good practice requires that it be set freshly each time a JWT is created
- **iat** - Issued At - This timestamp allows potential validation that the JWT is no more than a short time period old, for additional security hardening.
- **Tachyon.upn** - and body optional field (not shown in the example above) that allows you to specify which Principal you would like to authenticate as in cases where a single certificate can be used for multiple Principals.

Each of these members is known as a 'claim'. There is another reserved claim which is not included in this JWT and not required by IdPs normally. This is 'nbf' which is a timestamp before which the JWT is not valid.

The standard also allows custom claims to be added. There is a custom claim that we add if you want to be a specific principal (needed if the kid mappings allow it), which is **Tachyon.upn**. The code sample at the bottom mentions where to add it.

For the **aud** parameter, an external integration script can determine the IdP token issuing endpoint by making a GET request to the *<server>/api/Authentication/MetadataEndpoint endpoint*. This will return the openId metadata endpoint for the configured IdP. This endpoint is unsecured and allows any IdP consumer to determine all of the endpoints that the IdP supports - the token issuing endpoint will be among these. The integration script, having retrieved the metadata endpoint, queries this to in turn retrieve the token issuing endpoint it needs to fill in the JWT it is going to submit.

Once you have the JWT you should sign it using appropriate certificate. You can do that programmatically using whatever programming language of your choice. Here is an example for how you would do it using PowerShell:

```
function GetSignedClientAssertion($jwt, $certificate)
{
```

```
$jwtHdr = getJwtHdr (ToBase64Url $certificate.GetCertHash())
$jwtHdrJson = convertto-json $jwtHdr
$jwtHdrBytesEncoded = ToBase64UrlString $jwtHdrJson

$jwtJson = convertto-json $jwt
$jwtBytesEncoded = ToBase64urlString $jwtJson

$token = $jwtHdrBytesEncoded + "." + $jwtBytesEncoded
return GetSignedAssertion $token $certificate
}

function GetSignedAssertion($assertion, $certificate)
{
$rsa = [System.Security.Cryptography.X509Certificates.RSACertificateExtensions]
::GetRSAPrivateKey($certificate)
$assertionBytes = [System.Text.Encoding]::UTF8.GetBytes($assertion)
$padding = [System.Security.Cryptography.RSASignaturePadding]::Pkcs1
$signature = $rsa.SignData($assertionbytes, "SHA256", $padding)
$signatureEncoded = ToBase64Url $signature
$signedClientAssertion = $assertion + "." + $signatureEncoded
return $signedClientAssertion
}
```

Once you have a signed JWT assertion, you should make a POST request to:

https://your.tachyon.server/api/Authentication/RequestJwtAuthentication

And provide the assertion as the body of the request.

Remember that contrary to tokens obtained via interactive authentication, tokens obtained through JWT non-interactive authentication cannot be refreshed. Normally you will submit a JWT with an expiry time one hour into the future and you will receive a token with a lifetime of approximately one hour (this can be subject to variations depending on the IdP used and its settings).

Because the token you obtained using JWT cannot be refreshed, when it expires you need to repeat the cycle, generating a new JWT and resubmitting it to the RequestJwtAuthentication endpoint to acquire a new token.

Note

If you wrap token retrieval in a function that checks that the token is still valid, you can make this process transparent to the rest of your code, because then the retrieval function can automatically perform the request for a fresh token on behalf of your code.

2.5.1.4.4 Non-interactive authentication - Principal to certificate mapping

This page discusses how to manage certificate to principal mapping using the PowerShell Integration Toolkit.

2.5.1.4.4.1 About certificate to principal mapping

Platform-neutral authentication relies on certificates for external integration subsystems to authenticate to the 1E Platform. These certificates are used to sign Json Web Tokens (JWTs) which can then be verified by the IdP.

The 1E platform allows certificates to be mapped to Principals. A set of PowerShell cmdlets allows the user to set up and manage mapping entries.

Please note while you could use another language to perform this task as it is essentially all about sending right data to certain API endpoints, it is strongly advised to use Powershell toolkit published by 1E as it contains all the cmdlets required to perform these tasks.

2.5.1.4.4.2 Connecting to the platform

You cannot use certificate authentication to connect to the target platform until you have set up principal mappings. Consequently, to set these up you will connect to the target platform using interactive authentication. You will be prompted for credentials by the configured platform Identity Provider. These must correspond to a user in the platform with the appropriate privileges to manage mappings - typically, a full platform administrator.

To connect to the target platform, run the *set-1eserver* cmdlet from the PowerShell toolkit. You must first ensure that the toolkit is loaded. You need only do this once in a session.

Assuming you are in the folder into which the toolkit files were copied:

```
import-module .\ps1etoolkit.psd1 -force
```

Then connect to the platform server:

```
set-1Eserver <server name>
```

You can then run the cmdlets discussed below to set up certificate mappings. For example, *auth4ui2.platformeng.cloud.1e.com* You can then run the cmdlets discussed below to set up certificate mappings.

You use the *get-1ecertificatethumbprint* to list all certificates in a specified store. The 'Kid' property is the thumbprint represented in base-64 URL-safe encoding. You use this property to represent a certificate when setting up a mapping.

```
PS C:\source\Resources\DynamicScripts> get-tachyoncertificatethumbprint -storename localmachine\my

Subject                                 Thumbprint                                Kid
-------                                 ----------                                ---
CN=1EPRDROOTCA                          F94C4A766CE3B3FEF11555A76986E2D45A2C77FA  -UxKdmzjs/7xFVWnaYbi1Fosd/o
CN=x.y.z                                F3A22ADE21BB5D93245973D83859A6A0CC24DF86  86Iq3iG7XZMkWXPYOFmmoMwk34Y
CN=MyCert                               ED8A0A83D27A7813C179D5CDBC17AFF2AC4B7C47  7YoKg9J6eBPBedXNvBev8qxLfEc
CN=AnyClientInIISLab                    DD96C8DE756A217145FACB907AEF9D7E24CBA3E7  3ZbI3nVqIXFF-suQeu-dfiTLo-c
CN=*.platformeng.cloud.1e.com           C8EE931FEDF531679FB65B526C25A299A951470D  yO6TH-31MWeftltSbCWimalRRw0
CN=URTH-URTH-DC-CA, DC=URTH, DC=local   C59F9146F4319457438D1F05E3CCE499538CA977  xZ-RRvQxlFdDjR8F48zkmVOMqXc
CN=test code signing do not trust       A42AB4D93B8E72099748A5AF8EC0891AAEE843E4  pCq02TuOcgmXSKWvjsCJGq7oQ-Q
CN=p.q.r                                96A6BA2B73E0E45BBA247435F96825B9BDE027E7  lqa6K3Pg5Fu6JHQ1-Wglub3gJ-c
CN=1EPRDCAUK01, DC=1e, DC=local         5660A2017DFE6DCF74F6F8C4F1679F84570FA7AD  VmCiAX3-bc909vjE8WefhFcPp60
CN=URTH-DEV.URTH.local                  4D5E86DDF1A9C1224EC61C9538EC43C248C846C1  TV6G3fGpwSJOxhyVOOxDwkjIRsE
CN=localhost                            3B559A7E13EA654577FC8C7EA9AD0F16948648D3  O1WafhPqZUV3/Ix-qa0PFpSGSNM
O=1E, L=Ealing, S=London, C=GB          323048C57C2A82DF852E1BC9AAA2DC2793E3D2B1  MjBIxXwqgt-FLhvJqqLcJ5Pj0rE
CN=Corellian                            2D8C12E4DE7C5F39B90244D0766FA572965D1523  LYwS5N58Xzm5AkTOdm-lcpZdFSM
CN=URTH-URTH-SQL-CA, DC=URTH, DC=local  24FA7FB1D79831BBE1B09C07BF5EF01F1E07EE36  JPp/sdeYMbvhsJwHv17wHx4H7jY
CN=x.y.z                                03A72AA94471633D1C425C5C7FBD184DBD5A9B70  A6cqqURxYz0cQlxcf70YTb1am3A
```

We can see here the certificate we want to set up a mapping for and its 'Kid' property.

If you are generating your own JWT tokens, you can use any arbitrary string for the 'kid' field. Although you would typically set it to match the certificate thumbprint, the 1E platform does not actually require this. When creating the mapping, use whatever value you are using for your kid field, regardless of whether it matches the certificate thumbprint or not.

Note

You must be authenticated as a principal with appropriate rights in order to perform these operations, typically Full Administrator.

You use the *add-1ejwtprincipalmapping* cmdlet to add a mapping. We map the certificate highlighted above to the principal *Darth.Vader@corellianengineering.onmicrosoft.com*.

```
PS C:\source\Resources\DynamicScripts> add-tachyonjwtprincipalmapping  -identifier LYwS5N58Xzm5AkTQdm-lcpZdFSM
 -principal Darth.Vader@corellianengineering.onmicrosoft.com

Id Kid                          Upn
-- ---                          ---
 3 LYwS5N58Xzm5AkTQdm-lcpZdFSM Darth.Vader@corellianengineering.onmicrosoft.com
```

Having set up this mapping, any time an external integration subsystem uses the above certificate to authenticate, it will retrieve a token for the above principal and have all the permissions that the principal has in the 1E platform environment.

Note

You cannot add groups as principals. You must add specific user principals. However a user principal need not be directly represented in the 1E platform RBAC system. If any group or groups of which the principal is a member exists in the platform's RBAC database, then the permissions of the group or groups are applied.

RBAC checks are made when the signed JWT is presented for validation, not at the time of creating a mapping. Therefore, authentication may fail if the principal you have mapped to does not have rights in the platform's RBAC either directly or via group membership, at the time of authentication.

The *get-1Ejwtprincipalmapping* cmdlet returns all active certificate to principal mappings.

The *remove-1Ejwtprincipalmapping* cmdlets allows a mapping to be removed.

2.5.1.4.4.3 Advanced principal mapping options

You can specify a wildcard principal name (*) instead of a specific principal. This will allow the external integration subsystem to specify what user principal it wishes to act as, when authenticating. It does this by adding a 'Tachyon.upn' member to the body of the JWT it submits.

Warning

Do not use this option in production environments. If the associated certificate were exfiltrated, it could be used to authenticate as any user on the platform. This is, however, a handy setting in lab environments as it lets you easily authenticate as any principal, non-interactively, which can be handy for test purposes.

You can map the same certificate to multiple principals by adding it more than once but with a different principal name each time. Again, this allows the external integration subsystem to specify what user principal it wishes to act as. However, in this case it can only specify one of the principals which have been added.

2.5.1.4.4.4 Verifying mappings

You can use the 1E PowerShell toolkit to verify that your mappings are correctly set up. To do this, you need the application Id of the Identity Provider app configured to perform authentication and the subject of the signing certificate to be used. Your account on the endpoint on which you are running must have access to this certificate in the local machine certificate store, and it must have rights to retrieve the private key of this certificate.

> **Note**
>
> The application Id you require will be configured in the tenant's configuration on the platform.
>
> This is found either in the ClientAssertionAppId setting, or, if this is empty, the DirectoryAppId setting.
>
> You should also be able to find this Id in your IdP as it will be the Id of the application you have configured for JWT authentication.

If you have set up a single mapping row to a fixed principal, then you can connect with the command:

```
set-1Eserver -AppId <appid> -Certsubject <cert subject>
```

You can add the optional parameter:

```
-informationaction continue
```

if you want to receive detailed logging information as the operation is performed.

If you have set up multiple mapping rows for the same certificate, associating it with multiple principals, then you can connect with the command:

```
set-1Eserver -AppId <appid> -Certsubject <cert subject> -Principal <principal>
```

Where *<principal>* corresponds to one of the principals you have set up in the mapping table.

You can view the returned token with:

```
get-1Eauthtoken -decode
```

2.5.1.4.4.5 Using a dedicated identity provider application for external integration

The installation process currently assumes that you will share the identity provider application that the platform uses for directory search operations, for external integration.

In some cases, you may prefer to set up a dedicated identity provider application for external integration. This allows to restrict the permissions of that application. If a consumer were to obtain an identity provider bearer token directly from the directory search application, that bearer token would allow the consumer to read users, profiles and groups.

If you wish to control these permissions, they can set up a third application in their identity provider. They do this exactly as they do for the directory search application, but they can then choose that this application has only the default permissions, or customise accordingly.

When a JWT is submitted to the 1E platform, the application identity to which it will be forwarded within the identity provider is checked against two settings in the tenant configuration.

Normally only one of these settings is configured. In the example below, the directoryappid setting corresponds to the application id which is used by the platform for directory searches. It will also be shared for external integration.

```
"DirectoryAppId": "0c83091d-fc2f-40b5-8aca-940e4cf5f9b5",
.........
"ClientAssertionAppId": null,
```

When the JWT is submitted, the 1E platform confirms that the application id in the JWT matches the DirectoryAppId setting and then, if so, forwards the JWT to the identity provider for validation.

This check is made to prevent an attacker from submitting a JWT signed by an identity provider they control, instead of the expected identity provider.

If the platform did not verify the application Id, it would be possible to forward the JWT to an external attacker-controlled identity provider which would validate it, and then in turn the platform, believing the JWT to be valid, would return an internal token to the caller.

2.5.1.4.4.6 Specifying a dedicated identity provider application for external integration

To specify a separate application for external integration, you should set the ClientAssertionAppId member of the tenant's configuration to the separate application Id.

That way the JWTs submitted to the platform are checked against this application Id. Only JWTs specifying the new application Id are then passed to the identity provider. A JWT which specifies the shared directoryappid value will now be rejected and cannot be used for external integration.

Note

It is strongly recommended that you set up a dedicated identity provider application if possible. Doing so means that the bearer token returned by this application has only minimal privileges.

If you share the directory search application for external integration, then the bearer token returned by the identity provider will have the rights to read your user and group information from the identity provider (although it will not be able to change this information).

An external caller who can negotiate the oAuth grant flow directly with the identity provider can therefore obtain this bearer token directly. When you configure a dedicated application for external integration, you need not give the application any privileges as the bearer token returned is only used by the platform to verify the caller's identity. This means that any external caller who obtains this bearer token will have no privileges within the identity provider itself.

2.5.1.4.5 Implementing non-interactive (JWT) authentication - C# example

The code sample below is a complete Console application that compiles under the .NET Framework version 4.8.

Note

This is simplified for illustration. It is not production-ready code. Among other things, it lacks any error-checking for simplicity.

When the code does “Base64” it is actually doing what is commonly called “Url-Safe Base64”. This replaces some characters used by standard Base64 because they are not compatible with the character set allowed in a URL.

2.5.1.4.5.1 Before starting

Remember to configure the four constants at the top:

ApplicationId is the identifier for the Application that you added under your Azure portal for the purpose of doing non-interactive authentication. For testing or preproduction, you can use here the same application Id that you are using for Directory Access. However, for production you will want to use here a separate application with no permissions on your Azure Active Directory.

MetadataEndpoint is the endpoint in Azure that you call for getting the metadata for your application.

TachyonServer is the base address of your Tachyon server. Only the server – do not append the name of the Site and do not add a slash at the end.

CertificateThumbprint is the Thumbprint of the certificate that you are using to authenticate the calls made by your external application into Tachyon. This is the same certificate whose public key you have authorised to access ApplicationId. Remember to grant access on the private key to the user who will be executing the sample application. Also, remember that it needs to be mapped to a Tachyon user through the KidToUpnMapping. This sample program does not allow mapping one certificate to multiple users; to do this, you need to add the Tachyon.upn member to the JWT (see the comment in the code).

```
namespace ExternalIntegrationUsingJwt
{
    using System;
    using System.Net;
    using System.Net.Http;
    using System.Security.Cryptography;
    using System.Security.Cryptography.X509Certificates;
    using System.Text;
    using System.Threading;
    using System.Web.Script.Serialization;

    internal class Program
    {
        private const string TachyonConsumer = "your consumer name here"; // Yo
ur registered Consumer should go here
        private const int ExpirationSeconds = 3600;

        // Configure the following values:
        private const string ApplicationId = "8983091d-fc2f-40b5-8aca-940e4cf5f
9b5";
        private const string MetadataEndpoint = "https://login.microsoftonline.
com/89f2b2d5-56db-4d75-86d1-b64974bd35bd/v2.0/.well-known/openid-configuration"
;
        private const string TachyonServer = "https://your.tachyon.server";
        private const string CertificateThumbprint = "998c12e4de7c5f39b90244d07
```

```
66fa5b2965d1523";

        static void Main(string[] args)
        {
            // This is the code that obtains a Tachyon Token
            string signedJwt = GetSignedJwt();
            //string tachyonAuthEndpoint = TachyonServer + "/Tachyon/api/Authen
tication"; // for versions earlier than 24.1
            string tachyonAuthEndpoint = TachyonServer + "/api/Authentication";
            string tachyonToken = PostPayloadToTachyonAuthEndPoint(Quotify(sign
edJwt), tachyonAuthEndpoint);
            Console.WriteLine($"Got TachyonToken: {tachyonToken}");

            // This is an example for how we would use the token to invoke a Ta
chyon API
            //string exampleApi = TachyonServer + "/Tachyon/api/Permissions/Who
AmI"; // for versions earlier than 24.1
            string exampleApi = TachyonServer + "/api/Permissions/WhoAmI";
            string response = GetFromApi(exampleApi, tachyonToken);
            Console.WriteLine($"\nResponse from API: {response}");
        }

        private static string GetSignedJwt()
        {
            X509Certificate2 cert = GetCertFromLocalStore(CertificateThumbprint
);
            string upn = null; // This version is not currently adding the UPN
to the JWT
            string tokenEndpoint = GetTokenEndpoint();
            return GetAuthToken(cert, upn, tokenEndpoint, ApplicationId);
        }

        private static string GetAuthToken(X509Certificate2 cert, string upn, s
tring tokenEndpoint, string appId)
        {
            var token = GetJwt(tokenEndpoint, appId, appId, upn);
            var signedToken = GetSignedClientAssertion(token, cert);
            return signedToken;
        }

        private static string GetSignedClientAssertion(string jwtJson, X509Cert
ificate2 certificate)
        {
            var jwtHdrJson = GetJwtHdr(ToBase64Url(certificate.GetCertHash()));
            var jwtHdrBytesEncoded = ToBase64UrlString(jwtHdrJson);

            var jwtBytesEncoded = ToBase64UrlString(jwtJson);

            var token = jwtHdrBytesEncoded + "." + jwtBytesEncoded;
```

```
            return GetSignedAssertion(token, certificate);
        }

        private static string GetSignedAssertion(string assertion, X509Certific
ate2 certificate)
        {
            var rsa = RSACertificateExtensions.GetRSAPrivateKey(certificate);
            var assertionBytes = System.Text.Encoding.UTF8.GetBytes(assertion);
            var signature = rsa.SignData(assertionBytes, HashAlgorithmName.SHA2
56, RSASignaturePadding.Pkcs1);
            var signatureEncoded = ToBase64Url(signature);
            var signedClientAssertion = assertion + "." + signatureEncoded;
            return signedClientAssertion;
        }

        private static string GetJwtHdr(string thumbprint)
        {
            string jwtHdr = "{" + $"\"alg\":\"RS256\",\"typ\":\"JWT\",\"x5t\":\
"{thumbprint}\",\"kid\":\"{thumbprint}\"" + "}";
            return jwtHdr;
        }

        private static string GetJwt(string audience, string issuer, string sub
ject, string upn)
        {
            var timestamp = (ToUnixTimestamp(DateTime.UtcNow));
            var exptimestamp = timestamp + ExpirationSeconds;
            string jwtBody = "{" + $"\"aud\":\"{audience}\",\"iat\":\"{timestam
p}\",\"exp\":\"{exptimestamp}\",\"iss\":\"{issuer}\",\"sub\":\"{subject}\",\"jt
i\":\"{Guid.NewGuid()}\"" + "}";
            // Add "Tachyon.upn" if using a single certificate for multiple use
rs
            return jwtBody;
        }

        private static string ToBase64Url(byte[] buf)
        {
            var base64buf = Convert.ToBase64String(buf);
            base64buf = base64buf.Replace("=", "");
            base64buf = base64buf.Replace("+", "-");
            base64buf = base64buf.Replace("/", "_");
            return base64buf;
        }

        private static string ToBase64UrlString(string buf)
        {
            var bufBytes = Encoding.UTF8.GetBytes(buf);
            return ToBase64Url(bufBytes);
        }
```

```
        private static long ToUnixTimestamp(DateTime datetime)
        {
            var epoch = new DateTime(1970, 1, 1, 0, 0, 0);
            var seconds = Math.Truncate(datetime.ToUniversalTime().Subtract(epo
ch).TotalSeconds);
            return Convert.ToInt64(seconds);
        }

        private static string Quotify(string s)
        {
            return $"\"{s.Replace("\\", "\\\\").Replace("\"", "\\\"")}\"";
        }

        private static string PostPayloadToTachyonAuthEndPoint(string payload,
string tachyonAuthEndpoint)
        {
            using (HttpClient client = new HttpClient())
            {
                client.DefaultRequestHeaders.Add("X-Tachyon-Consumer", TachyonC
onsumer);

                string endpoint = tachyonAuthEndpoint + "/RequestJwtAuthenticat
ion";
                var result = client.PostAsync(endpoint, new StringContent(paylo
ad, Encoding.UTF8, "application/json"), CancellationToken.None).Result;
                string response = result.Content.ReadAsStringAsync().Result;
                return response;
            }
        }

        private static X509Certificate2 GetCertFromLocalStore(string thumbprint
)
        {
            X509Certificate2 cer = null;

            // Assume certs are always in Personal store of local machine
            using (var store = new X509Store("My", StoreLocation.LocalMachine))
            {
                store.Open(OpenFlags.ReadOnly);
                var cers = store.Certificates.Find(X509FindType.FindByThumbprin
t, thumbprint, false);
                if (cers.Count > 0)
                {
                    cer = cers[0];
                }
                else
                {
                    store.Close();
                    throw new Exception($"Could not find certificate with thumb
print={thumbprint}");
```

```
                }

                store.Close();
            }

            return cer;
        }

        private static string GetFromApi(string endpoint, string tachyonToken)
        {
            using (HttpClient client = new HttpClient())
            {
                client.DefaultRequestHeaders.Add("X-Tachyon-Consumer", TachyonC
onsumer);
                client.DefaultRequestHeaders.Add("X-Tachyon-Authenticate", tach
yonToken);

                var result = client.GetAsync(endpoint, CancellationToken.None).
Result;
                string response = result.Content.ReadAsStringAsync().Result;
                return response;
            }
        }

        private static string GetTokenEndpoint()
        {
            var wc = new WebClient();
            string metadata = wc.DownloadString(MetadataEndpoint);
            JavaScriptSerializer js = new JavaScriptSerializer();
            MetaData deserialized = js.Deserialize<MetaData>(metadata);

            return deserialized.token_endpoint;
        }

        private class MetaData
        {
            public string token_endpoint { get; set; }
        }
    }
}
```

2.5.1.4.6 Implementing non-interactive (JWT) authentication - Python example

This page shows some sample Python code to perform non-interactive authentication to the platform. This code will sign a JWT, present it to the platform, receive an authentication token and then call the /devices endpoint to retrieve a list of all devices that are known to the platform as an example API call.

2.5.1.4.6.1 Setting up the environment

We recommend using Conda to set up a Python environment. As this is well-documented elsewhere, this is not covered here. You should use Python 3.8 or later. Once Conda and Python are installed, use conda's install option to add the required modules which you can see in the example code.

> **Note**
>
> The *jwt* module is called *pyjwt*.

> **Note**
>
> While this material is covered on the parent page, it is included here for convenience, so that if you are implementing a Python integration project everything you need to know is on this page.

To run this example code, it is assumed that you have access to a platform server. You must have defined an external integration application Id on the associated identity provider instance. You need a self-signed certificate, which you can create by using PowerShell e.g

```
new-selfsignedcertificate -subject CN=Integration
```

There is no significance to the subject, it can be anything you wish. This will create the certificate in the local machine personal certificate store.

Export the certificate using the certlm.msc utility and specify that the private key is to be exported. Specify a password. In the sample code this is *Password*. You will export to a .PFX file. In the sample code this file is called *integrate.pfx*

> **Note**
>
> In a SaaS environment you can choose to retrieve the certificate from the Azure Key Vault. A snippet of Python code to do this is included in the appendix below. In the example code, the certificate is retrieved directly from the PFX file.

Export the certificate a second time but this time specify not to include the private key. This should create a .CER file with just the public key. Ensure that the public key is uploaded to the application on your identity provider that is used for external integration.

In the example code below, this application Id is configured in the line:

```
idpAppServiceClientId = "0c830a1d-fc2f-40b5-8aca-940e4cf5f9b5"
```

Change this to match the application Id of the application that is configured on your identity provider.

> **Note**
>
> The platform does not require that this application have any permissions on the identity provider itself.

Ensure that the platform server is configured to use this application for external integration. This corresponds to setting the ClientAssertionAppId property in its tenantconfig.json file as shown below, where the application Id is the above configured external integration app id.

```
......
"ClientAssertionAppId": "0c830a1d-fc2f-40b5-8aca-940e4cf5f9b5",
......
```

Configure the sample code to connect to the platform server.

```
platformServer = "https://your.platform.server"
```

The platform needs to know the principal that the JWT will map to, since it will issue a token for this principal. To set up the mapping:

Use the 1E PowerShell toolkit to interactively authenticate to the platform using a principal with Full Administrator rights. In the example code the platform server is https://tachyonrts.urth.local so the command will be:

```
set-1eserver your.platform.server
```

Get the 'kid' value for the JWT signing certificate by using the cmdlet:

```
get-1ecertificatethumbprint -storename localmachine\my
```

Use the add-1ejwtprincipalmapping cmdlet to add a mapping

```
add-1ejwtprincipalmapping
```

When prompted for an identifier, paste in the 'kid' from the previous step.

When prompted for a principal, enter the principal name that you wish the kid to map to.

> **Note**
>
> With this single mapping, the example code as written will acquire a token for that principal.
>
> As noted in the code, it is possible to map a single certificate kid to multiple principals if desired. If you do this, then the custom JWT claim that is commented out in the example code must be included so that the calling code can indicate to the platform which principal it wishes to act as.
>
> Note also that a single identity provider application configured for external integration can alternatively have multiple certificate public keys uploaded to it. In that scenario, you set up mappings as above for each 'kid' associated with those certificates. In this case each certificate maps independently to one or more principals. This gives you great flexibility in setting up external integration subsystems and managing their permissions.

Once you have completed this step, you should be able to run the sample code. A screenshot of an example lab run is included below.

```
import jwt
import datetime
```

```
import base64
import hashlib
import uuid
from cryptography.hazmat.primitives.serialization import pkcs12
import json
from cryptography import x509
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.hazmat.primitives.asymmetric import padding
import requests

import warnings
warnings.filterwarnings("ignore")  # this is just to suppress the warnings abou
t overriding SSL trust for a lab platform where Python's cert store doesn't hav
e a CA root cert imported.


TrustPlatformServer = False #allows us to call a platform server (e.g in a lab)
where the Python cert store doesn't have a CA root cert establishing an SSL tru
st chain. Otherwise set to True

# This just loads the cert from a local Pfx file. Alternatively you can load th
e cert from the Azure Key Vault

def LoadCertFromPfx(pfx, password):
    with open(pfx, "rb") as pfxFile:
        pfxBinData = pfxFile.read()
        return pkcs12.load_key_and_certificates(pfxBinData, password)

# convert to URL-safe encoding

def toBase64Url(input):
    return base64.urlsafe_b64encode(input).rstrip(b'=')

# get certificate thumbprint (which we will then convert to a key identifier (k
id) for the JWT later by base 64 url-safe encoding it)

def calculateThumbprint(certificate):
    der = certificate.public_bytes(serialization.Encoding.DER)
    digest = hashes.Hash(hashes.SHA1(), backend=default_backend())
    digest.update(der)
    thumbprint = digest.finalize()
    return thumbprint

# get the configured metadata endpoint for the identity provider from the platf
orm

def getMetadataEndpoint(platformMetadataEndpoint):
    return requests.get(platformMetadataEndpoint, verify=TrustPlatformServer).j
son()
```

```
# get the identity provider token issuing endpoint from its openId metadata end
point
# Note that the identity provider will always present a trusted SSL cert that P
ython's cert store can resolve to a trust chain, so we don't need to have an ov
erride for this

def getTokenEndpoint(identityProviderMetadataEndpoint):
    return requests.get(identityProviderMetadataEndpoint).json()["token_endpoin
t"]


# create a signed JWT using the pfx file with the certificate (this could be re
trieved from the Azure Key Vault instead), the pfx password, the client asserti
on audience and the configured IdP application Id that will verify the JWT
# (this must have the public key of this cert uploaded to it)

def generateSignedJwt(pfx, password, clientAssertionAudience, idpAppServiceClie
ntId):
    privateKey, certificate, _ = LoadCertFromPfx(pfx,password)
    kid = toBase64Url(calculateThumbprint(certificate)).decode()

    now = datetime.datetime.utcnow()
    expirationTime = now + datetime.timedelta(seconds=3600)

    header = {
        'alg': 'RS256',
        'typ': 'JWT',
        'x5t': kid,
        'kid': kid
    }

# You do not need a claim for the servicePrincipalName unless the platform kid-
>upn mapping table has multiple entries for the same kid (i.e certificate thumb
print)
# Otherwise the mapping is implied to a specific principal.

    body = {
        'aud': clientAssertionAudience,
        'iat': now,
        'exp': expirationTime,
        'iss': idpAppServiceClientId,
        'sub': idpAppServiceClientId,
        'jti': str(uuid.uuid4()) #,
#        'Tachyon.upn' : servicePrincipleName    # this is a custom claim allow
ing one certificate to map to multiple principals on the platform
    }
    signedJwt = jwt.encode(body, key=privateKey, algorithm='RS256', headers=hea
der)
```

```
    return signedJwt

# make a request to the platform to get a token (base-64 encoded)

def requestJwtAuthentication(platformConsumer, platformAuthEndpoint, payload):
    headers = {"X-Tachyon-Consumer": platformConsumer}
    endpoint = platformAuthEndpoint + "/RequestJwtAuthentication"
    response = requests.post(endpoint, headers=headers, json=payload, verify=Tr
ustPlatformServer)
    responseContent = response.text
    return responseContent


# Example API call

def getFromApi(platformServer, platformConsumer, path, token):
    headers = {"X-Tachyon-Consumer": platformConsumer, "X-Tachyon-Authenticate"
: token}
    endpoint = platformServer + path
    response = requests.get(endpoint, headers=headers,verify=TrustPlatformServe
r)
    return response.json()


# Main code here.

platformServer = "https://your.platform.server"
platformConsumer = "your registered consumer"
platformAuthEndpoint = platformServer + "/api/Authentication"
platformMetadataEndpoint = platformAuthEndpoint + "/metadataendpoint"

# This is the application Id of the app that is set up for external integration
. Your certificate's public key must have been uploaded to that app

idpAppServiceClientId = "0c830a1d-fc2f-40b5-8aca-940e4cf5f9b5"


# Get the configured identity provider's metadata endpoint from the platform

metadataEndpoint = getMetadataEndpoint(platformMetadataEndpoint)
print(metadataEndpoint)

# call the openId metadata endpoint on the identity provider to get the identit
y provider's token issuing endpoint

tokenEndpoint = getTokenEndpoint(metadataEndpoint)
print(tokenEndpoint)


# Make a signed JWT using the example PFX file.
```

```
signedJwt = generateSignedJwt("integrate.pfx",b'Password', tokenEndpoint, idpAp
pServiceClientId)


# Request an internal token from the platform. This will have a life of one hou
r. When it expires, you repeat the above step to get a fresh signed Jwt and the
n request a fresh token

internalAccessToken = requestJwtAuthentication(platformConsumer, platformAuthEn
dpoint, signedJwt)

print("Response:", internalAccessToken)

# Call an API, passing the token (in this case, getting a list of connected dev
ices)
apiResult = getFromApi(platformServer, platformConsumer, "/consumer/devices", i
nternalAccessToken)
print(apiResult)
```

```
# Install the required libraries
!pip install azure-keyvault-secrets azure-identity

# Import the necessary modules
from azure.identity import ClientSecretCredential
from azure.keyvault.secrets import SecretClient

# Create a secret client with client secret authentication
credential = ClientSecretCredential(tenant_id, client_id, client_secret)
secret_client = SecretClient(vault_url=key_vault_url, credential=credential)

# Retrieve the secret value
cert_base64 = secret_client.get_secret(secret_name).value

# Use the secret value in your code
print("Retrieved secret value:", cert_base64)
```

2.5.1.4.7 Implementing non-interactive (JWT) authentication - Typescript example

This page shows a sample implementation of non-interactive authentication to the 1E Platform using TypeScript.

Please note that this is simplified for illustration. It is not production-ready code. Among other things, it lacks any error-checking for simplicity. This code was written to run inside the 'bun' environment, which is basically similar to node.js, so there should be equivalent library functionality in both environments.

> **Note**
>
> This example retrieves the JWT signing certificate from a local PFX file.
>
> This does not represent optimum security best practice. As with any implementation, regardless of language, there are other mechanisms available to retrieve certificates. For example:

- From the local Windows certificate store (where applicable).
- From a mounted Kubernetes secret.
- From the Azure Credential Vault.

Obtaining the certificate from any of those sources is outside the bounds on this article.

```
import { createHash } from 'crypto'
import fs from 'fs'

import { PlatformMetadata } from '@e-types'
import { sign } from 'jsonwebtoken'
import { asn1, pkcs12, pki, util } from 'node-forge'

const pfxPath = 'security/development-1e-vault-idp-1e-20240213.pfx'

const getTokenEndpoint = async () => {
  const url = await fetch('https://login.microsoftonline.com/a6c15a4f-bd3b-4eac
-aac9-ef6e9ce805b0/v2.0/.well-known/openid-configuration')
  const metadataEndpoint = await url.json()
  return (metadataEndpoint as PlatformMetadata).token_endpoint
}

const toBase64Url = input => {
  return input.toString('base64url')
}

const calculateThumbprint = der => {
  const sha1 = createHash('sha1')
  sha1.update(der)
  return toBase64Url(sha1.digest())
}

const loadCertFromPfx = () => {
  const pfxData = fs.readFileSync(pfxPath)
  const pfxAsBinary = util.decode64(pfxData.toString('base64'))
  const der = asn1.fromDer(pfxAsBinary)
  const pfx = pkcs12.pkcs12FromAsn1(der, false)

  const keyBags = pfx.getBags({ bagType: pki.oids.pkcs8ShroudedKeyBag })
  const certBags = pfx.getBags({ bagType: pki.oids.certBag })
  const privateKey = pki.privateKeyToPem(keyBags[pki.oids.pkcs8ShroudedKeyBag][
0].key)
  const certificate = certBags[pki.oids.certBag][0].cert

  const certDer = asn1.toDer(pki.certificateToAsn1(certificate as unknown as pk
i.Certificate)).getBytes()
  const thumbprint = calculateThumbprint(Buffer.from(certDer, 'binary'))

  return { privateKey, thumbprint }
}
```

```
export const generateJwt = async () => {
  const tokenEndpoint = await getTokenEndpoint()
  const today = new Date()
  const appId = '95ba1ee7-de87-4add-82f4-e900f13bc57f'
  const { privateKey: key, thumbprint: kid } = loadCertFromPfx()
  const now = Math.floor(today.getTime() / 1000)
  const expirationTime = now + 3600

  const payload = {
    aud: tokenEndpoint,
    iat: now,
    exp: expirationTime,
    iss: appId,
    sub: appId,
    jti: crypto.randomUUID(),
  }

  const header = {
    alg: 'RS256',
    x5t: kid,
    kid: kid,
  }

  const signedJwt = sign(payload, { key, passphrase: '' }, { algorithm: 'RS256'
, header })
  const url = await fetch('https://your.tachyon.server/api/Authentication/Reque
stJwtAuthentication', {
    method: 'POST',
    headers: { 'X-Tachyon-Consumer': 'Explorer', 'Content-Type': 'application/j
son' },
    body: JSON.stringify(signedJwt),
  })
  return await url.text()
}
```

2.5.2 Add and remove Instruction Definitions and organize them into Instruction Sets

This section covers uploading instruction definitions into Tachyon and organizing them into sets.

2.5.2.1 Instruction Definitions

Instruction definitions are templates, used by Tachyon to issue Instructions. Before using an instruction definition it has to be uploaded and assigned to an instruction set.

2.5.2.1.1 Uploading Instruction Definitions

Instruction definitions can be uploaded individually, one XML at a time, or in bulk as a zip file containing multiple instruction definitions:

Note

When using this method to POST an instruction, the format of the POST payload is a MIME multipart document. The C# SDK contains helper functions to handle this automatically. If you are working in another environment, such as PowerShell, then the payload format looks like this for a raw, uncompressed XML file

1. You must add a header to the request indicating the MIME type and boundary to the POST request.

```
Content-Type: multipart/form-data; boundary=xyzzy
```

(where xyzzy is an arbitrary delimiter string you do not expect to find in your payload content). A GUID is a good choice

2. The payload is then constructed by concatenating the MIME multipart header as shown below. In the sample, the uploaded filename is x.xml

```
--xyzzy

Content-Disposition: form-data; name="x.xml"; filename="x.xml"
Content-Type: text/xml

<the XML that corresponds to the signed instruction i.e contents of the XML fil
e>
--xyzzy--
```

Note that the MIME delimiter begins with two hyphens at the start and has two hypens at the front AND end as the closing delimiter

For a zipped file the content will be set to application/zip

Direct Consumer API call	C# code using Consumer SDK library
Make a POST request to https://my.tachyon.server/Consumer/ProductPacks attach the file (either XML or zip) to the request. The return payload contains information about instruction definitions found in the file and at the status of the upload. **Return payload** Expand source `[{ "InstructionDefinitionName": "1E-Explorer-Examples-CountEventLogErrorsInLastX", "InstructionSetName": "Unassigned",`	Use ProductPacks object inside the Tachyon connector instance. **Uploading instruction definitions** Expand source `response = connector.ProductPacks.Upload("c:\\product packs\\1E-ConfigMgrConsoleExtensions.zip");` response.ReceivedObject object contains the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "Status": "New"
},
{
    "InstructionDefinitionName": "1E-Exp
lorer-Examples-DeleteCertificates",
    "InstructionSetName": "Unassigned",
    "Status": "New"
}]
```

Upload instruction definitions into an existing instruction set:

Direct Consumer API call	C# code using Consumer SDK library
Make a POST request to https://my.tachyon.server/Consumer/ProductPacks/InstructionSet/Id/1 and attach the file (either XML or zip) to the request. The return payload will (as in the above example) contain information about instruction definitions found in the file and at the status of the upload. **Return payload** Expand source `[{ "InstructionDefinitionName": "1E-Explorer-Examples-CountEventLogErrorsInLastX", "InstructionSetName": "Hardware questions", "Status": "New" }, { "InstructionDefinitionName": "1E-Explorer-Examples-DeleteCertificates", "InstructionSetName": "Hardware questions", "Status": "New" }]`	Use ProductPacks object inside the Tachyon connector instance. **Uploading instruction definitions** Expand source `response = connector.ProductPacks.UploadIntoInstructionSet("c:\\product packs\\1E-ConfigMgrConsoleExtensions.zip", 1);` response.ReceivedObject object contains the same data you can see in the JSON response on the left.

2.5.2.1.2 Deleting Instruction Definitions

An instruction definition can be deleted if there are no pending, in progress or scheduled instructions based on it. If there are, either stop those instructions, let them run their course or delete the schedule.

Delete a single instruction definition, using either its Id or name:

Direct Consumer API call	C# code using Consumer SDK library

To delete an instruction definition using its Id make a DELETE request to https://my.tachyon.server/Consumer/InstructionDefinitions/Id/1
 	
To delete an instruction definition using its name make a DELETE request to https://my.tachyon.server/Consumer/InstructionDefinitions/ name/1E-Explorer-Examples-CountEventLogErrorsInLastX
	Use InstructionDefinitions object inside the Tachyon connector instance.
	
	Delete an Instruction Definition by its Id
	
	`connector.InstructionDefinitions.DeleteInstructionDefinition(1);`
	
	Delete an Instruction Definition by its Name
	
	`connector.InstructionDefinitions.DeleteInstructionDefinition("1E-Explorer-Examples-CountEventLogErrorsInLastX");`

Delete multiple instruction definitions in a single call using a list of either Ids or names:

Direct Consumer API call	C# code using Consumer SDK library

To delete multiple instruction definitions using their Ids make a DELETE request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionDefinitions Expand source
 	
`{ "Ids": [ 1,2 ]}`
 	
To delete multiple instruction definitions using their names make a DELETE request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionDefinitions Expand source
 	
`{ "Names": [ "1E-Explorer-Examples-CountEventLogErrorsInLastX", "1E-Explorer-Examples-DeleteCertificat`
	Use InstructionDefinitions object inside the Tachyon connector instance. To delete instruction definition using their Ids:
	
	Deleting multiple instruction definitions by their Ids Expand source
	
	`var payload = new List<int> { 1, 2 }; var response = connector.InstructionDefinitions.DeleteInstructionDefinitions(payload);`
	
	To delete instruction definitions using their names:
	
	Deleting multiple instruction definitions by their name Expand source

Direct Consumer API call	C# code using Consumer SDK library

es"
]}
	var payload = new List<string>
	{
	"1E-Explorer-Examples-CountEventLogErrorsInLastX",
	"1E-Explorer-Examples-DeleteCertificates"
	};
	var response = connector.InstructionDefinitions.DeleteInstructionDefinitions(payload);

2.5.2.2 Instruction Sets

Instruction sets are containers holding instruction definitions:

- With permissions assigned to instruction sets, not to individual instructions
- Containing any number of instruction definitions and cannotbe nested
- Instruction definitions can onlybe assignedto one instruction set, or none at all.

An instruction definition cannot belong tomultipleinstruction sets at once.

It also does not have to belong to a set.If an instruction definition does not belong to an instruction set, you cannot issue instructionsbasedon that definition.

2.5.2.2.1 Add an Instruction Set

To add an instruction set all you need is a name. This name needs to be unique and no two instruction sets can have the same name.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
Payload sent to https://my.tachyon.server/Consumer/InstructionSets
Expand source
{
"Name": "Hardware questions"
}
Returns the following response:
Return payload Expand source
	Use InstructionSets object inside the Tachyon connector instance.
	Creating an Instruction Set
	Expand source
	var payload = new Tachyon.SDK.Consumer.Models.Send.InstructionSet
	{
	Name = "Hardware questions"
	};

Direct Consumer API call	C# code using Consumer SDK library

<pre>{
    "Id": 1,
    "Name": "Hardware questions",
    "Description": null,
    "Icon": null
}</pre>
	<pre>var response = connecto
	r.InstructionSets.Add(p
	ayload);</pre>
	response.ReceivedObject object will contain the same data you can see in the JSON response on the left.

An instruction set can also have a description and an icon.

The icon can be supplied either directly as binary data inside the Icon parameter or as the same binary data encoded in a base64 string supplied in IconInBase64 parameter.

UseIcon parameter is a flag indicating if an icon should be used.

If this flag is set to true, icon will be used in following way, if:

- Icon field (byte array) is supplied, it will be used as the icon
- Icon isn't supplied and IconInBase64 is supplied, it will be used as the icon
- both Icon and IconInBase64 are supplied, Icon will be used and IconInBase64 will be ignored
- Neither Icon or IconInBase64 were supplied, icon will not be used and if there already was an icon assigned to the instruction set, it will be removed.

If this flag is set to false, no changes will be made to the icon. Both Icon and IconInBase64 fields are ignored.

To create an Instruction Set with a description and an icon:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionSets
Expand source
 	
<pre>{
  "Name": "Hardware questions",
  "Description": "This is a description",
  "UseIcon": true,
  "Icon": [<binary data here>]
}</pre>
	Use InstructionSets object inside the Tachyon connector instance.
	
	To create an instruction set that uses an icon supplied as a binary array you can use code below and supply the icon inside binaryData variable.

Direct Consumer API call	C# code using Consumer SDK library

will create an instruction set that uses binary data inside Icon. In order to use binary data encoded as base64 string make POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/InstructionSets
Expand source

```
{
  "Name": "Hardware questions",
  "Description": "This is a description",
  "UseIcon": true,
  "IconInBase64": "<binary data encoded as base64 he
re>"
}
```

Both POST calls will yield following response because the icon is always returned in its binary form:

Return payload Expand source

```
{
    "Id": 1,
    "Name": "Hardware questions",
    "Description": "This is a description",
    "Icon": [<binary data here>]
}
```

Creating an Instruction Set
Expand source

```
var payload = new Tachy
on.SDK.Consumer.Models.
Send.InstructionSet
{
    Name = "Hardware qu
estions",
    Description = "This
is a description",
    UseIcon = true,
    Icon = binaryData
};

var response = connecto
r.InstructionSets.Add(p
ayload);
```

To create an instruction set that uses an icon supplied as binary data encoded in a base64 string, you can use code below and supply the icon inside encodedData variable:

Creating an Instruction Set
Expand source

```
var payload = new Tachy
on.SDK.Consumer.Models.
Send.InstructionSet
{
    Name = "Hardware qu
estions",
    Description = "This
is a description",
    UseIcon = true,
    IconInBase64 = enco
dedData
};

var response = connecto
```

Direct Consumer API call	C# code using Consumer SDK library
	`r.InstructionSets.Add(payload);` In either case, response.ReceivedObject object will contain the same data you can see in the JSON response on the left.

2.5.2.2.2 Edit an Instruction Set

To edit an instruction set, you'll need its Id.

You can modify all properties of an instruction set, apart from the Id. Icon follows the same rules as outlined in the previous paragraph. This means the payload looks similar to that used to create an instruction set, with the addition of Id, and making PUT requests instead of POST.

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionSets Expand source
 	
`{ "Id": 1, "Name": "Hardware instructions" }`
 	
Returns this response providing the instruction set with the Id of 1 didn't have an icon:
 	
Return payload Expand source
 	
`{ "Id": 1, "Name": "Hardware instructions", "Description": null, "Icon": null }`
	Use InstructionSets object inside the Tachyon connector instance.
	
	Updating an Instruction Set Expand source
	
	`var payload = new Tachyon.SDK.Consumer.Models.Send.InstructionSet { Id = 1, Name = "Hardware instructions" };`
	
	`var response = connector.InstructionSets.Update(payload);`
	
	response.ReceivedObject object will contain the same data you can see in the JSON response on the left.

2.5.2.2.3 Delete an Instruction Set

An instruction set can be deleted, causing all instruction definitions belonging to it to become unassigned.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/InstructionSets/Id/ 1	Use InstructionSets object inside the Tachyon connector instance. **Deleting an Instruction Set** `connector.InstructionSets.Delete(1);`

You can also delete an instruction set along with all instruction definitions inside that set. Use this option with care as it permanently removes the instruction definitions.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/InstructionSets/Id/ 1?deleteContent=true	Use InstructionSets object inside the Tachyon connector instance. **Deleting an Instruction Set with all Instruction Definitions in that set** `connector.InstructionSets.DeleteWithContents(1, true);`

You cannot delete Instruction Definitions with pending, in progress Instructions or scheduled Instructions, so trying to delete an Instruction Set with contents with instruction definitions like these will fail.

2.5.2.2.4 Get Instruction Sets

The first example shows how to retrieve all available instruction sets:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionSets returns this response: **Return payload** Expand source	Use InstructionSets object inside the Tachyon connector instance: **Retrieving all Instruction Sets** `response = connector.InstructionSets.GetAll();`

Direct Consumer API call	C# code using Consumer SDK library

<pre>[{
    "Id": 1,
    "Name": "Hardware instructions",
    "Description": "This pack contains questio
ns and actions dealing with hardware",
    "Icon": null
},{
    "Id": 2,
    "Name": "Software instructions",
    "Description": "This pack contains questio
ns and actions dealing with software",
    "Icon": null
},{
    "Id": 3,
    "Name": "Patch success instructions",
    "Description": null,
    "Icon": null
}]</pre>
	response.ReceivedObject object will contain the same data you can see in the JSON response on the left.

You can also retrieve all instruction sets along with the count of instruction definitions that belong to each set:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/InstructionSets returns this response:
 	
Return payload Expand source
 	
<pre>[{
    "Id": 1,
    "Name": "Hardware instructions",
    "Description": "This pack contains questio
ns and actions dealing with hardware",
    "Icon": null,
    "CountOfDefinitions": 22
},{
    "Id": 2,
    "Name": "Software instructions",
    "Description": "This pack contains questio
ns and actions dealing with software",
    "Icon": null,
    "CountOfDefinitions": 37
},{
    "Id": 3,
    "Name": "Patch success instructions",
    "Description": null,</pre>
	Use InstructionSets object inside the Tachyon connector instance:
	
	Retrieving all Instruction Sets
	
	<pre>response = connector.Instruct
	ionSets.GetAllWithCounts();</pre>
	response.ReceivedObject object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "Icon": null,
    "CountOfDefinitions": 8
}]
```

You can also retrieve a specific instruction set by its Id or Name:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/InstructionSets/Id/1 returns this response:
 	
Return payload Expand source
 	
`{`
`  "Id": 1,`
`  "Name": "Hardware questions",`
`  "Description": null,`
`  "Icon": null`
`}`
	Use InstructionSets object inside the Tachyon connector instance:
	
	Retrieving specific Instruction Set
	
	`response = connector.InstructionSets.Get(id);`
	
	response.ReceivedObject object will contains the same data as the JSON response on the left.

2.5.2.2.5 Add Instruction Definitions to an Instruction Set

Instruction definitions can be added to an Instruction set either using the Instruction Set API or Instruction Definitions API.

Using the Instruction Set API:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionSets/Contents Expand source
 	
`{`
`  "SetId": 1,`
`  "InstructionDefinitionIds": [`
`  1,2,3`
`  ]`
`}`
 	
returns this response:
 	
Return payload Expand source
	Use InstructionSets object inside the Tachyon connector instance.
	
	Adding Instruction Definitions to specific Instruction Set
	
	`response = connector.InstructionSets.AddInstructionDefinitions(1, new List<int> {1,2,3});`
	
	response.ReceivedObject object contains the same data you as the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
{
    "Id": 1,
    "Name": "Hardware questions",
    "Description": null,
    "Icon": null
}
```

If you do not provide SetId in your payload, the call will succeed but the instructions will become unassigned.

Using Instruction Definition API:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/InstructionDefinitions/Id/1/InstructionSet/1 without any payload returns this response: **Return payload** Expand source `{ "Id": 1, "Name": "1E-Explorer-Examples-CountEventLogErrorsInLastX", "Description": "Returns event log entries aggregated on event ID (EventCode) of a specified type in the a specified Windows event log in the last few days. Specifying a long period may result in a large amount of data from each agent.", "InstructionSetId": 2, "InstructionSetName": "Wszystko", "InstructionType": 0, "ReadablePayload": "Which event IDs of type %type% have been logged in the Windows %logname% event log in the last %numDays% days?", "Parameters": [{ "Name": "type", "Pattern": "%type%", "DataType": "string", "ControlType": "valuePicker", "ControlMetadata": null, "Placeholder": "event type", "DefaultValue": null, "Validation": {`  Use InstructionDefinitions object inside the Tachyon connector instance.  **Addign specific Instruction Definition to an Instruction Set**  `response = connector.InstructionDefinitions.AddDefinitionToSet(1, 1);` response.ReceivedObject object contains the same data in the JSON response on the left.

```
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": ["information",
            "warning",
            "error"],
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    },
    {
        "Name": "logname",
        "Pattern": "%logname%",
        "DataType": "string",
        "ControlType": "valuePicker",
        "ControlMetadata": null,
        "Placeholder": "log name'",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": ["application",
            "security",
            "setup",
            "system"],
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    },
    {
        "Name": "numDays",
        "Pattern": "%numDays%",
        "DataType": "int",
        "ControlType": "freeText",
        "ControlMetadata": null,
        "Placeholder": "number of days",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": null,
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
```

```
        "Source": null
    }],
    "Schema": [{
        "Name": "Category",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "CategoryString",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "EventCode",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "EventIdentifier",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "Message",
        "Type": "String",
        "Length": 4096,
        "RenderAs": null
    },
    {
        "Name": "RecordNumber",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "SourceName",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "GeneratedTimeStamp",
        "Type": "datetime",
        "Length": 0,
```

```
        "RenderAs": null
    },
    {
        "Name": "User",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    }],
    "Aggregation": {
        "Schema": [{
            "Name": "EventCode",
            "Type": "int64",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "Count",
            "Type": "int32",
            "Length": 0,
            "RenderAs": null
        }],
        "GroupBy": "EventCode",
        "Operations": [{
            "Name": "Count",
            "Type": "count"
        }]
    },
    "InstructionTtlMinutes": 30,
    "ResponseTtlMinutes": 30,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 10080,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
    "ResponseTemplateId": 1,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [{
            "Id": "leftpie",
            "Title": "Number of EventIds acr
oss devices",
            "Type": "Pie",
            "X": "EventCode",
            "Y": "Count",
            "PostProcessor": "processingFunc
tion",
            "Size": 1,
```

Direct Consumer API call	C# code using Consumer SDK library

```
            "Row": 1
        }],
        "PostProcessors": [{
            "Name": "processingFunction",
            "Function": "ProcessSingleSeries
('EventCode', 'Count', '9', 'Number of event
s', 'true')"
        }]
    },
    "Version": "3",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2019-03-19T10:1
4:32.227Z",
    "NumberOfTimesExecuted": 0,
    "Executable": true
}
```

The InstructionSet API allows you to bulk assign Definitions, while InstructionDefinitions API allows you to only assign one definition to a set at a time.

2.5.2.2.6 Remove Instruction Definitions from an Instruction Set

Instruction definitions can be removed from an Instruction set either using the Instruction Set API or Instruction Definitions API.

Using Instruction Set API:

Direct Consumer API call	C# code using Consumer SDK library

Make a DELETE request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionSets/Contents Expand source
 	
`{`
`  "SetId": 1,`
`  "InstructionDefinitionIds": [`
`  1,2,3`
`  ]`
`}`
	Use InstructionSets object inside the Tachyon connector instance:
	
	Removing Instruction Definitions from specific Instruction Set
	
	`connector.InstructionSets.RemoveInstructionDefinitions(1, new List<int> { 1, 2, 3 });`

Using Instruction Definition API.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer /InstructionDefinitions/Id/1/InstructionSet/1	Use InstructionDefinitions object inside the Tachyon connector instance: **Removing a specific Instruction Definition from specific Instruction Set** `connector.InstructionDefinitions.RemoveDefinitionFromSet(1, 1);`

2.5.2.2.7 Export Instruction Definitions from a set

The following example exports all instruction definitions that belong to a specific set, allowing you to download an entire Instruction Set without knowing which instruction definitions belong to that set.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer /InstructionSets/Id/{id}/Export will give you back a binary stream with a zip archive that contains all Instruction Definitions from the requested set.	Use InstructionSets object inside the Tachyon connector instance: **Exporting all Instruction Definitions that belong to a specific Instruction Set** `response = connector.InstructionSets.Export(1);` response.ReceivedObject will be a MemoryStream holding a zip file containing Instruction Definitions.

2.5.3 Find the right instruction definition

This section covers how to browse available instruction definitions in your Tachyon installation.

In the following examples, the instruction definitions have alreadybeen uploadedinto Tachyon, organized into Instruction Sets and have assigned permissions.For more information about this read Add and remove Instruction Definitions and organize them into Instruction Sets and Set up Principals, Roles and Permissions, up to and including Tachyon version 5.2.

The C# examples assume you're using Tachyon Consumer SDK and you have an instantiated instance of Tachyon connector class in an object called connector.

All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API. In the following examples this detailis leftout, stating that the returned object contains the data. For example, when we say that XYZ object contains certain data, this means the ReceivedObject contains that data, since that is always true.

When this section mentions permissions on instructions, it means instruction setpermissions not Instruction set managementpermissions. Instruction set management permissions are a separate category of permissions. They gain Principals access to all instructions within a specific set or all sets.

It's possible to have a permission on specific, or every instruction set and be able to issue those instructions without being able to manage their instruction set (that is, without having associated instruction set management permissions).

The reverse is also possible and the Principal that can assign instruction definitions to an instruction set may not have the necessary permissions to issue any of them.

2.5.3.1 What is an instruction definition?

An instruction definition is a template used to create instructions which are sent to computers running Tachyon Agent for processing.

The instruction definition defines what the agent does when executing the instruction, for example what data is returned, if it is aggregated, what the parameters for the instruction are, or how to present the results to the user.

When an instruction is issued, some information is copied from the definition the instruction is based on (like instruction name and type) while others come from the information provided with the request to issue the instruction. For more information read issue an instruction, track its progress and retrieve responses.

2.5.3.1.1 What properties can be used to uniquely identify an instruction definition?

An instruction definition has two properties to identify it, Id and name:

Id - is a number assigned to the instruction definition when it's imported into Tachyon. This means you cannot predict the Id before the instruction is imported and it will be different between Tachyon installations. If you remove the instruction definition and import it again it will be given a different Id.

Name - must be unique in each Tachyon installation. Importing a new instruction definition with the same name as one already in Tachyon is treated as an upgrade of the existing instruction definition.

The API allows you to use both Id and Name when locating specific instruction definitions or issuing an instruction.

How to uniquely identify an instruction?

Instructions can only be identified by their Id, which is assigned when an instruction is created (issued). They cannot be located by Name, because it's not unique. Every instruction based on the same instruction definition will have the same name.

2.5.3.1.2 How is an instruction created from an instruction definition

Full details of this process are outside the scope of this section.

When an instruction is created (sometimes called issuing) the:

- Instruction definition name gets copied into the newly created instruction
- The payload and readable payload get copied into the newly created instruction - any parameters (provided by the issuer of the instruction), environment variables and global settings are completed
- Instruction TTL and Response TTL are filled with values provided by the instruction issuer

- Schema and Aggregation schema get copied from the instruction definition
- Scope provided by the instruction issuer is combined with any compulsory scope that results from issuing Principal permissions.

Note

This is not a complete list.

2.5.3.2 Searching for instruction definitions

Instruction definitions are retrieved for two reasons:

Issuing instructions - instruction set related permissions of the Principal have to be taken into account. Those permissions limit which instructions the caller is allowed to issue. This limitation is enforced by the API endpoint.

Managing instruction sets - the Principal retrieving them requires permission to manage instruction sets and does not need to have any permissions to the instructions themselves.

This is important because if you retrieve all instruction definitions using the endpoint designed for instruction set management, you might not be able to issue all of those instructions. This is because the list you receive will not be filtered down using your permissions.

2.5.3.2.1 Retrieving all instruction definitions

This example retrieves all instruction definitions that can be issued:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions **Response received from** https://my.tachyon.server/Consumer/InstructionDefinitions Expand source `[{ "Id": 1, "Name": "1E-Explorer-Examples-CountEventLogErrorsInLastX", "Description": "Returns event log entries aggregated on event ID (EventCode) of a specified type in the a specified Windows event log in the last few days. Specifying a long period may result in a large amount of data from each agent.", "InstructionSetId": 2, "InstructionSetName": "Wszystko", "InstructionType": 0, "ReadablePayload": "Which event IDs of type %type% have been logged in the Win`	Use InstructionDefinitions object inside the Tachyon connector instance. **Retrieving all instruction definitions the caller has permissions to** Expand source `var definitions = connector.InstructionDefinitions.Get(null);` Definitions object will contain the same data you can see in the JSON response on the left.

```
dows %logname% event log in the last %num
Days% days?",
    "Parameters": [{
        "Name": "type",
        "Pattern": "%type%",
        "DataType": "string",
        "ControlType": "valuePicker",
        "ControlMetadata": null,
        "Placeholder": "event type",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": ["informatio
n",
            "warning",
            "error"],
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    },
    {
        "Name": "logname",
        "Pattern": "%logname%",
        "DataType": "string",
        "ControlType": "valuePicker",
        "ControlMetadata": null,
        "Placeholder": "log name'",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": ["applicatio
n",
            "security",
            "setup",
            "system"],
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    },
    {
        "Name": "numDays",
        "Pattern": "%numDays%",
```

```
        "DataType": "int",
        "ControlType": "freeText",
        "ControlMetadata": null,
        "Placeholder": "number of days",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": null,
            "AllowedValues": null,
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    }],
    "Schema": [{
        "Name": "Category",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "CategoryString",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "EventCode",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "EventIdentifier",
        "Type": "int64",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "Message",
        "Type": "String",
        "Length": 4096,
        "RenderAs": null
    },
    {
        "Name": "RecordNumber",
        "Type": "int64",
```

```
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "SourceName",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "GeneratedTimeStamp",
        "Type": "datetime",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "User",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    }],
    "Aggregation": {
        "Schema": [{
            "Name": "EventCode",
            "Type": "int64",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "Count",
            "Type": "int32",
            "Length": 0,
            "RenderAs": null
        }],
        "GroupBy": "EventCode",
        "Operations": [{
            "Name": "Count",
            "Type": "count"
        }]
    },
    "InstructionTtlMinutes": 30,
    "ResponseTtlMinutes": 30,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 1008
0,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
```

```
    "ResponseTemplateId": 1,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [{
            "Id": "leftpie",
            "Title": "Number of EventIds
across devices",
            "Type": "Pie",
            "X": "EventCode",
            "Y": "Count",
            "PostProcessor": "processingF
unction",
            "Size": 1,
            "Row": 1
        }],
        "PostProcessors": [{
            "Name": "processingFunction",
            "Function": "ProcessSingleSer
ies('EventCode', 'Count', '9', 'Number of
events', 'true')"
        }]
    },
    "Version": "3",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2019-03-19T1
0:14:32.227Z",
    "NumberOfTimesExecuted": 0,
    "Executable": true
},
{
    "Id": 2,
    "Name": "1E-Explorer-Examples-DeleteC
ertificates",
    "Description": "Deletes all certifica
tes with a given SHA1 thumbprint in the '
local machine' certificate store for thes
e stores: Personal (My), Intermediate Cer
tification Authorities (CA), Trusted Root
Certification Authorities (Root), Third-P
arty Certification Authorities (AuthRoo
t), Enterprise Trust (Trust), Other Peopl
e (AddressBook), Trusted People (TrustedP
eople), Trusted Publishers (TrustedPublis
her) and Untrusted Certificates (Disallow
ed). Certificates are not retrievable onc
e marked for deletion. Please use with ca
```

```
ution - deleting a certificate may case a
number of things to stop working.",
    "InstructionSetId": 2,
    "InstructionSetName": "Wszystko",
    "InstructionType": 1,
    "ReadablePayload": "Delete all certif
icates with a SHA1 thumbprint of %thumbpr
int% from the 'local machine' certificate
store.",
    "Parameters": [{
        "Name": "thumbprint",
        "Pattern": "%thumbprint%",
        "DataType": "string",
        "ControlType": "freeText",
        "ControlMetadata": null,
        "Placeholder": "thumbprint",
        "DefaultValue": null,
        "Validation": {
            "Regex": null,
            "MaxLength": "256",
            "AllowedValues": null,
            "NumValueRestrictions": null
        },
        "Value": null,
        "HintText": null,
        "Source": null
    }],
    "Schema": [{
        "Name": "Subject",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "Thumbprint",
        "Type": "String",
        "Length": 128,
        "RenderAs": null
    },
    {
        "Name": "StoreName",
        "Type": "String",
        "Length": 256,
        "RenderAs": null
    },
    {
        "Name": "DeletionTime",
        "Type": "String",
```

Direct Consumer API call	C# code using Consumer SDK library


```
        "Length": 256,
        "RenderAs": null
    }],
    "Aggregation": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 120,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 1008
0,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
    "ResponseTemplateId": null,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": nul
l,
    "Version": "3",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2019-03-19T1
0:14:32.227Z",
    "NumberOfTimesExecuted": 0,
    "Executable": true
}]
```


The payload has been trimmed to just two entries to show what you can expect. In a production installation of Tachyon there will be many more instruction definitions, perhaps hundreds.

Retrieving only instruction definitions of a specific type

In some cases, you may want to retrieve all instruction definitions of a given type that you can issue. This is used, for example, by Tachyon Explorer on the Follow-up question and Follow-up action tabs on the instruction details page. You can do this using the optional query string parameter on the same call as the one made in the last example.

The next example shows where we only get the Questions we're allowed to issue.

Direct Consumer API call	C# code using Consumer SDK library
Make a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions?instructionType=question	Use InstructionDefinitions object inside the Tachyon connector instance. **Retrieving all instruction definitions of a specific type that the caller is allowed to issue** Expand source

Direct Consumer API call	C# code using Consumer SDK library
The return payload will have the same structure as the payload seen when we retrieved all the instruction definitions in the previous example.	`var definitions = connector.InstructionDefinitions.Get(new List<InstructionType>{InstructionType.Question});` Definitions object contains information similar to the retrieving all instruction definitions example.

Retrieving all instruction definitions regardless of caller permissions

Retrieving all definitions should only be used when managing instruction sets.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/All The return payload has same structure as the payload seen when retrieving all the instruction definitions in the retrieving all instruction definitions examples.	Use InstructionDefinitions object inside the Tachyon connector instance. **Retrieving all instruction definitions in the system, regardless of caller's permissions** Expand source `var definitions = connector.InstructionDefinitions.GetAllInstructionDefinitions();` Definitions object contains information similar to retrieving all instruction definitions examples.

Please note that this endpoint requires Instruction set management related permissions to work.

2.5.3.2.2 Using search-as-you-type feature

This is the feature seen on the Tachyon Explorer home page as "I want to know" in form of a search box where Tachyon returns the instruction definitions you have permissions to use based on your search criteria.

The match is achieved by searching for the words (defined as a string of characters separated by a space) in an instruction definition's readable payload (which is what Explorer displays after you've picked an instruction definition) and description (which Explorer displays in the tooltip available after an instruction definition is selected).

This example looks for the word bios and retrieves all instruction definitions containing that word either in the description or readable payload.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/Search/Ymlvcw== returns results similar to:	Use InstructionDefinitions object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
Response received from https://my.tachyon.server/Consumer/InstructionDefinitions/Search Expand source `{ "Result": 0, "Error": null, "Instructions": [{ "Id": 66, "Name": "1E-Explorer-TachyonCore-BiosDetails", "ReadableName": "What BIOS firmware is installed?", "Description": "Returns details of BIOS firmware.", "Category": "SomeCategory", "WordMatchCount": 1, "InstructionType": 0, "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 60, "ParameterHints": null, "Aggregatable": true, "IsLicensed": true, "NumberOfTimesExecuted": 0, "Executable": null }] }`	**Retrieving instruction definitions matching a search term** Expand source `var definitions = connector.InstructionDefinitions.FindInstructions("bios");` Definitions object contains the same data you can see in the JSON response on the left.

Entries for Instruction definitions returned by this endpoint contain only part of the data for performance reasons. This is intended as a search-as-you-type feature so the data must be returned quickly.

2.5.3.2.3 Searching for instruction definitions that match specific criteria

When managing instruction sets you can also use an endpoint allowing instruction definitions to be filtered, sorted and paginated. This endpoint, like the one allowing you to retrieve all instructions regardless of calling Principal's permissions, requires permissions related to instruction set management to work.

This example looks for all instruction definitions containing the word bios in their name. Although it looks the same as the search-as-you-type scenario, it looks exclusively on the Name, and not on the readable payload or description because our search parameters only specify the name.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to	Use InstructionDefinitions object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

https://my.tachyon.server/Consumer /InstructionDefinitions/Search with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/InstructionDefinitions/Search Expand source
 	
`{ "Start": 1, "PageSize": 200, "Filter": { "Attribute": "Name", "Operator": "LIKE", "Value": "%bios%" } }`
 	
Returns a similar response to the one below:
 	
Return payload Expand source
 	
`{ "TotalCount": 1, "Items": [{ "Id": 66, "Name": "1E-Explorer-TachyonCore-BiosDetails", "Description": "Returns details of BIOS firmware.", "InstructionSetId": 2, "InstructionSetName": "Wszystko", "InstructionType": 0, "ReadablePayload": "What BIOS firmware is installed?", "Parameters": null, "Schema": [{ "Name": "BIOSVersion", "Type": "string", "Length": 512, "RenderAs": null }, { "Name": "Caption", "Type": "string",`
	Retrieving filtered, sorted and paginated instruction definitions Expand source
	
	`var settings = new Search { PageSize = 200, Start = 1, Filter = new ExpressionObject { Attribute = "Name", Operator = "LIKE", Value = "%bios%" } }; var definitions = connector.InstructionDefinitions.FindInstructions(settings);`
	
	Definitions object contain the same data you can see in the JSON response on the left.

```
                    "Length": 512,
                    "RenderAs": null
                },
                {
                    "Name": "Manufacturer
",
                    "Type": "string",
                    "Length": 512,
                    "RenderAs": null
                },
                {
                    "Name": "PrimaryBIOS
",
                    "Type": "bool",
                    "Length": 0,
                    "RenderAs": null
                },
                {
                    "Name": "ReleaseDate
",
                    "Type": "datetime",
                    "Length": 0,
                    "RenderAs": null
                },
                {
                    "Name": "SerialNumber
",
                    "Type": "string",
                    "Length": 512,
                    "RenderAs": null
                }
            ],
            "Aggregation": {
                "Schema": [
                    {
                        "Name": "Manufact
urer",
                        "Type": "string",
                        "Length": 512,
                        "RenderAs": null
                    },
                    {
                        "Name": "BIOSVers
ion",
                        "Type": "string",
                        "Length": 512,
                        "RenderAs": null
                    },
```

```
                    {
                        "Name": "Count",
                        "Type": "int64",
                        "Length": 0,
                        "RenderAs": null
                    }
                ],
                "GroupBy": "Manufacturer,
BIOSVersion",
                "Operations": [
                    {
                        "Name": "Count",
                        "Type": "count"
                    }
                ]
            },
            "InstructionTtlMinutes": 60,
            "ResponseTtlMinutes": 60,
            "MinimumInstructionTtlMinutes
": 10,
            "MaximumInstructionTtlMinutes
": 10080,
            "MinimumResponseTtlMinutes":
10,
            "MaximumResponseTtlMinutes":
10080,
            "Workflow": null,
            "ResponseTemplateId": 1,
            "OriginalFileName": null,
            "ResponseTemplateConfiguratio
n": {
                "Name": "default",
                "TemplateConfigurations":
[
                    {
                        "Id": "mainchart
",
                        "Title": "Most co
mmon BIOS manufacturers and versions",
                        "Type": "Bar",
                        "X": "BIOSVersion
",
                        "Y": "Count",
                        "Z": "Manufacture
r",
                        "PostProcessor":
"processingFunction",
                        "Size": 1,
```

Direct Consumer API call	C# code using Consumer SDK library

```
                        "Row": 1
                    }
                ],
                "PostProcessors": [
                    {
                        "Name": "processi
ngFunction",
                        "Function": "Proc
essMultiSeries('BIOSVersion', 'Count', 'M
anufacturer', '5', '5', 'false')"
                    }
                ]
            },
            "Version": "3",
            "Author": "1E",
            "IsLicensed": true,
            "UploadedTimestampUtc": "2019
-03-19T10:14:39.363Z",
            "NumberOfTimesExecuted": 0,
            "Executable": null
        }
    ]
}
```

If you want to search the same way as search-as-you-type feature, you can adjust the search criteria to include Readable payload and Description columns.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to

https://my.tachyon.server/Consumer/InstructionDefinitions/Search with following payload:

Payload sent to https://my.tachyon.server/Consumer/InstructionDefinitions/Search
Expand source

```
{
    "Start": 1,
    "PageSize": 200,
    "Filter": {
        "Operator": "OR",
        "Operands": [{
            "Attribute": "Name",
            "Operator": "LIKE",
```

Use InstructionDefinitions object inside the Tachyon connector instance.

Retrieving instruction definitions with the word "bios" in the name, description or readable payloyad
Expand source

```
var settings = new Search
{
    PageSize = 200,
    Start = 1,
    Filter = new ExpressionObject
    {
        Operator = "OR",
        Operands = new List<ExpressionObjec
t>
        {
            new ExpressionObject
```

Direct Consumer API call

```
            "Value": "%bios%"
        },
        {
            "Attribute": "Descri
ption",
            "Operator": "LIKE",
            "Value": "%bios%"
        },
        {
            "Attribute": "Readab
lePayload",
            "Operator": "LIKE",
            "Value": "%bios%"
        }]
    }
}
```

Returns a similar response to the one below:

Return payload Expand source

```
{
    "TotalCount": 1,
    "Items": [
        {
            "Id": 66,
            "Name": "1E-Explorer
-TachyonCore-BiosDetails",
            "Description": "Retu
rns details of BIOS firmware.",
            "InstructionSetId":
2,
            "InstructionSetName
": "Wszystko",
            "InstructionType":
0,
            "ReadablePayload": "
What BIOS firmware is installed?
",
            "Parameters": null,
            "Schema": [
                {
                    "Name": "BIO
SVersion",
                    "Type": "str
ing",
                    "Length": 51
```

C# code using Consumer SDK library

```
                {
                    Attribute = "Name",
                    Operator = "LIKE",
                    Value = "%bios%"
                },
                new ExpressionObject
                {
                    Attribute = "Description",
                    Operator = "LIKE",
                    Value = "%bios%"
                },
                new ExpressionObject
                {
                    Attribute = "ReadablePayloa
d",
                    Operator = "LIKE",
                    Value = "%bios%"
                }
            }
        }
};

var definitions = connector.InstructionDefi
nitions.FindInstructions(settings);
```

Definitions object contains the same data you can see in the JSON response on the left.

```
2,
                    "RenderAs":
null
                },
                {
                    "Name": "Cap
tion",
                    "Type": "str
ing",
                    "Length": 51
2,
                    "RenderAs":
null
                },
                {
                    "Name": "Man
ufacturer",
                    "Type": "str
ing",
                    "Length": 51
2,
                    "RenderAs":
null
                },
                {
                    "Name": "Pri
maryBIOS",
                    "Type": "boo
l",
                    "Length": 0,
                    "RenderAs":
null
                },
                {
                    "Name": "Rel
easeDate",
                    "Type": "dat
etime",
                    "Length": 0,
                    "RenderAs":
null
                },
                {
                    "Name": "Ser
ialNumber",
                    "Type": "str
ing",
                    "Length": 51
```

```
2,
                    "RenderAs":
null
                }
            ],
            "Aggregation": {
                "Schema": [
                    {
                        "Name":
"Manufacturer",
                        "Type":
"string",
                        "Length
": 512,
                        "RenderA
s": null
                    },
                    {
                        "Name":
"BIOSVersion",
                        "Type":
"string",
                        "Length
": 512,
                        "RenderA
s": null
                    },
                    {
                        "Name":
"Count",
                        "Type":
"int64",
                        "Length
": 0,
                        "RenderA
s": null
                    }
                ],
                "GroupBy": "Manu
facturer,BIOSVersion",
                "Operations": [
                    {
                        "Name":
"Count",
                        "Type":
"count"
                    }
                ]
```

```
            },
            "InstructionTtlMinut
es": 60,
            "ResponseTtlMinutes
": 60,
            "MinimumInstructionT
tlMinutes": 10,
            "MaximumInstructionT
tlMinutes": 10080,
            "MinimumResponseTtlM
inutes": 10,
            "MaximumResponseTtlM
inutes": 10080,
            "Workflow": null,
            "ResponseTemplateId
": 1,
            "OriginalFileName":
null,
            "ResponseTemplateCon
figuration": {
                "Name": "default
",
                "TemplateConfigu
rations": [
                    {
                        "Id": "m
ainchart",
                        "Title":
"Most common BIOS manufacturers
and versions",
                        "Type":
"Bar",
                        "X": "BI
OSVersion",
                        "Y": "Co
unt",
                        "Z": "Ma
nufacturer",
                        "PostPro
cessor": "processingFunction",
                        "Size":
1,
                        "Row": 1
                    }
                ],
                "PostProcessors
": [
                    {
```

Direct Consumer API call	C# code using Consumer SDK library

```
                        "Name":
"processingFunction",
                        "Functio
n": "ProcessMultiSeries('BIOSVer
sion', 'Count', 'Manufacturer',
'5', '5', 'false')"
                    }
                ]
            },
            "Version": "3",
            "Author": "1E",
            "IsLicensed": true,
            "UploadedTimestampUt
c": "2019-03-19T10:14:39.363Z",
            "NumberOfTimesExecut
ed": 0,
            "Executable": null
        }
    ]
}
```

2.5.3.3 Searching for instruction definitions when issuing a follow-up instruction

To issue an instruction as a follow up to another instruction, additional conditions need to be taken into account.

If you were to simply retrieve all definitions as in retrieving all instruction definitions, although you'd get a list of instruction definitions that you, the caller, have permission to, you'd ignore all restrictions coming from the instruction that's the parent (the instruction you're following up on). Those restrictions stem from permissions of whoever issued the instruction you're about to follow up on. You need to use another endpoint to make sure you're searching for instruction definitions in the right context.

2.5.3.3.1 Retrieving all instruction definitions that can be used to issue a follow-up

You can retrieve all definitions you have permission to, like with retrieving all instruction definitions, in the context of a follow-up instruction.

This example assumes you're following up on an instruction with the Id of 12. The instruction itself is irrelevant for this example which focuses on establishing what instruction definitions could be used for a follow-up.

Direct Consumer API call	C# code using Consumer SDK library
Make a GET request to https://my.tachyon.server/Consumer/Instruction Definitions/Parent/12 to retrieve	Use InstructionDefinitions object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
all instruction definitions the caller can issue as a follow-up. The return payload will have the same structure as the payload seen when we were retrieving all the instruction definitions in retrieving all instruction definitions. If you want to only get instruction definitions of a specific type, add a query string parameter as in retrieving only instruction definitions of a specific type. For instance, if you wanted to get all Actions you'd make this GET request: https://my.tachyon.server/Consumer/InstructionDefinitions/Parent/12?instructionType=action	**Retrieving all instruction definitions that can be used to issue follow up to instruction 12** Expand source `var definitions = connector.InstructionDefinitions.GetInstructionDefinitionsAvailableAsFollowUp(12);` The definitions object will contain the same data we've seen when retrieving all the instruction definitions in retrieving all instruction definitions.

2.5.3.3.2 Searching for feasible follow-up instruction definitions that match specific criteria

You can also use a search-as-you-type endpoint, like the using search-as-you-type feature, when looking for instruction definitions to issue follow-up instructions.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/Parent/12/Search/Ymlvcw== will return the following response:
 	
Response received from https://my.tachyon.server/Consumer/InstructionDefinitions/Search Expand source
 	
`{`
`"Result": 0,`
`"Error": null,`
`"Instructions": [`
	Use InstructionDefinitions object inside the Tachyon connector instance.
	
	Retrieving instruction definitions matching a search term that can be used to follow-up the instruction number 12 Expand source
	
	`var definitions = connector.InstructionDefinitions.GetInstructionDefinitionHintsForFollowUp("bios", 12, null);`

Direct Consumer API call	C# code using Consumer SDK library

<pre>{
"Id": 66,
"Name": "1E-Explorer-TachyonCore-BiosDetails",
"ReadableName": "What BIOS firmware is installed?",
"Description": "Returns details of BIOS firmware.",
"Category": "SomeCategory",
"WordMatchCount": 1,
"InstructionType": 0,
"InstructionTtlMinutes": 60,
"ResponseTtlMinutes": 60,
"ParameterHints": null,
"Aggregatable": true,
"IsLicensed": true,
"NumberOfTimesExecuted": 0,
"Executable": null
}
]
}</pre>
	The definitions object will contain the same data you can see in the JSON response on the left.

2.5.3.4 Retrieving specific instruction definition

You can also attempt to retrieve a specific definition, either by its Id or Name.

The following two examples look for the Bios instruction definition, both using its Id, which is 66 as seen in previous examples and its name which is 1E-Explorer-TachyonCore-BiosDetails.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions /Id/66 returns the following response:
 	
Response received from https://my.tachyon.server/Consumer/InstructionDefinitions/Id/66 Expand source
 	
<pre>{
"Id": 66,
"Name": "1E-Explorer-TachyonCore-BiosDetails",
"Description": "Returns details of BIOS fir</pre>
	Use InstructionDefinitions object inside the Tachyon connector instance.
	
	Retrieving a specific instruction definition by its Id Expand source
	
	<pre>var definition = connector.InstructionDefinitions.GetInstructionDefinition(66);</pre>

Direct Consumer API call	C# code using Consumer SDK library

mware.",
"InstructionSetId": 2,
"InstructionSetName": "Wszystko",
"InstructionType": 0,
"ReadablePayload": "What BIOS firmware is i
nstalled?",
"Parameters": null,
"Schema": [
{
"Name": "BIOSVersion",
"Type": "string",
"Length": 512,
"RenderAs": null
},
{
"Name": "Caption",
"Type": "string",
"Length": 512,
"RenderAs": null
},
{
"Name": "Manufacturer",
"Type": "string",
"Length": 512,
"RenderAs": null
},
{
"Name": "PrimaryBIOS",
"Type": "bool",
"Length": 0,
"RenderAs": null
},
{
"Name": "ReleaseDate",
"Type": "datetime",
"Length": 0,
"RenderAs": null
},
{
"Name": "SerialNumber",
"Type": "string",
"Length": 512,
"RenderAs": null
}
],
"Aggregation": {
"Schema": [
	The definition object will contain the same data you can see in the JSON response on the left.

```
            {
                "Name": "Manufacturer",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "BIOSVersion",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Count",
                "Type": "int64",
                "Length": 0,
                "RenderAs": null
            }
        ],
        "GroupBy": "Manufacturer,BIOSVersion",
        "Operations": [
            {
                "Name": "Count",
                "Type": "count"
            }
        ]
    },
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 10080,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
    "ResponseTemplateId": 1,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [
            {
                "Id": "mainchart",
                "Title": "Most common BIOS manu
facturers and versions",
                "Type": "Bar",
                "X": "BIOSVersion",
                "Y": "Count",
                "Z": "Manufacturer",
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "PostProcessor": "processingFun
ction",
                "Size": 1,
                "Row": 1
            }
        ],
        "PostProcessors": [
            {
                "Name": "processingFunction",
                "Function": "ProcessMultiSeries
('BIOSVersion', 'Count', 'Manufacturer', '5', '
5', 'false')"
            }
        ]
    },
    "Version": "3",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2019-03-19T10:14:3
9.363Z",
    "NumberOfTimesExecuted": 0,
    "Executable": null
}
```

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/Name/1E-Explorer-TachyonCore-BiosDetails returns the same response seen in the previous example.	Use InstructionDefinitions object inside the Tachyon connector instance. **Retrieving a specific instruction definition by its Name** Expand source `var definition = connector.InstructionDefinitions.GetInstructionDefinition("1E-Explorer-TachyonCore-BiosDetails");` The definition object will contain the same information it did in the previous example.

2.5.3.5 Exporting Instruction Definitions

You can export Instruction Definitions back into their original XML format.

>

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/Id/1/export gives you a binary stream with an XML document representing requested Instruction Definition.
 	
You can also export it by name by making a GET request to https://my.tachyon.server/Consumer/InstructionDefinitions/Name/1E-Explorer-Examples-CountEventLogErrorsInLastX/export
	Use InstructionDefinitions object inside the Tachyon connector instance:
	
	Exporting specific Instruction Definition by its Id Expand source
	
	`response = connector.InstructionDefinitions.ExportDefinition(1);`
	
	Or by name:
	
	Exporting specific Instruction Definition by its Name Expand source
	
	`response = connector.InstructionDefinitions.ExportDefinition("1E-Explorer-Examples-CountEventLogErrorsInLastX");`
	
	response.ReceivedObject will be a MemoryStream holding an XML file containing the requested Instruction Definition.

You can also export multiple Instruction Definitions at the same time, specifying either their Ids or Names, giving you a zip file containing one XML file for each Instruction Definition.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request with following payload if you want to export Instruction Definitions by Id: **Payload sent to** https://my.tachyon.server/Consumer/InstructionDefinitions/Export Expand source `{ "Ids": [ 1,2 ]}` Or this payload if you want to export them by name:	Use InstructionDefinitions object inside the Tachyon connector instance. **Exporting multiple Instruction Definitions** Expand source `response = connector.InstructionDefinitions.ExportDefinitions(new List<int> {1, 2});` response.ReceivedObject will be a MemoryStream containing a zip file containing one XML file

Direct Consumer API call	C# code using Consumer SDK library
Payload sent to https://my.tachyon.server/Consumer/InstructionDefinitions/Export Expand source { "Names": ["1E-Explorer-Examples-CountEventLogErrorsInLastX", "1E-Explorer-Examples-DeleteCertificates"]} Returns a byte stream with a zip file containing requested Instruction Definitions.	for each Instruction Definition. The Consumer SDK does not support exporting multiple Instruction Definitions by name.

2.5.4 Getting data from Tachyon clients

This section covers how to get information about computers connected to Tachyon. Tachyon keeps information about computers that are connected to it (through the Tachyon agent\client) or that connected to it within last 90 days (this is configurable).

Although from Tachyon 4.1 the Tachyon Agent is now called the 1E Client this section still refers to it as the Tachyon Agent. This is because we're dealing with the Tachyon part of the 1E Client.

This section helps you understand how to check what impact an instruction has on your estate.

In this section:

- The C# examples assume you're using the Tachyon Consumer SDK
- You have an instantiated instance of Tachyon connector class in an object called connector
- A computer is a device running the Tachyon Agent service
- All SDK methods return the same class called ApiCallResponse.

Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API.

In the following examples this detail is left out, stating that the returned object contains the data . For example, when we say that XYZ object contains certain data, this means the ReceivedObject contains that data, since that's always true.

2.5.4.1 Find all computers

The most basic use case is finding every computer Tachyon knows about:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Devices will return this response: **Response received from** https://my.tachyon.server/Consumer/Devices Expand source [{ "Id": 1, "Name": "Somemachine120", "Fqdn": "Somemachine120.somedomain.com", "Status": 0, "OsType": "Windows", "OsVerNum": 1688863374311424, "OsVerTxt": "Microsoft Windows 7 Enterprise", "AgentVersion": 281483566841860, "Manufacturer": "Dell Inc.", "ChassisType": 7, "DeviceType": "Desktop", "CpuType": "Intel(R) Xeon(R) CPU E5-2630 0 @ 2. 30GHz", "CpuArchitecture": "x64", "OsArchitecture": "x64", "RamMB": 32693, "SMBiosGuid": "55f144f2-c52b-4349-add4-1467dc34 449e", "TachyonGuid": "eadad09d-1dd0-4f72-966a-f53d130 2875f", "LastBootUTC": "2019-02-04T00:00:00Z", "LastConnUtc": "2019-03-07T08:14:35.93Z", "CreatedUtc": "2019-02-03T00:00:00Z", "VrPlatform": "", "TimeZone": 0, "CertType": "Domain", "CertExpiryUtc": "2019-10-09T04:53:24Z", "PushToken": null, "Model": "Precision T5600", "Domain": "SomeDomain", "Tags": "||", "ConnectionState": [], "LocalIpAddress": null, "TimeZoneId": null, "SerialNumber": null, "Criticality": 1, "CoverageTags": {}	Use Devices object inside the Tachyon connector instance and call GetDevices without any parameters. **Retrieving devices** Expand source `var devices = connector.Devices.GetDevices();` The devices object contains the same data as the JSON response on the left.

```
    },
    {
        "Id": 2,
        "Name": "Somemachine121",
        "Fqdn": "Somemachine121.somedomain.com",
        "Status": 0,
        "OsType": "Windows",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Microsoft Windows 7 Enterprise",
        "AgentVersion": 281483566841860,
        "Manufacturer": "Dell Inc.",
        "ChassisType": 7,
        "DeviceType": "Desktop",
        "CpuType": "Intel(R) Xeon(R) CPU E5-2630 0 @ 2.
30GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "81725c90-63b8-4e82-ac79-8ff19ffb
54a7",
        "TachyonGuid": "5c76c6fd-cda4-4690-ac1a-d394cb2
104f0",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Precision T5600",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    },
    {
        "Id": 3,
        "Name": "Somemachine122",
        "Fqdn": "Somemachine122.somedomain.com",
        "Status": 0,
        "OsType": "Windows",
        "OsVerNum": 1688863374311424,
```

```
        "OsVerTxt": "Microsoft Windows 7 Enterprise",
        "AgentVersion": 281483566841860,
        "Manufacturer": "Dell Inc.",
        "ChassisType": 7,
        "DeviceType": "Desktop",
        "CpuType": "Intel(R) Xeon(R) CPU E5-2630 0 @ 2.
30GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "b30efd3e-39e3-4c52-bf5e-cfc096d0
545f",
        "TachyonGuid": "eff2792d-83f3-411f-9e04-684f66a
c50ea",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Precision T5600",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    }
]
```


2.5.4.2 Finding a specific computer or computers

To find a specific computer or computers matching specific criteria, you should use a filter expression.

2.5.4.2.1 Filtering and sorting with pagination

The device's API supports standard filtering, sorting and pagination methods seen throughout the Tachyon Consumer API to find computers matching specific parameters. See API reference for a list of the columns you can sort and filter.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with the following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Devices Expand source
 	
{
  "Start": 1,
  "PageSize": 3,
  "Filter": {
  "Attribute": "OsType",
  "Operator": "==",
  "Value": "Linux"
  }
}
 	
Returns this response:
 	
Return payload Expand source
 	
{
  "TotalCount": 10,
  "Items": [
  {
  "Id": 41,
  "Name": "Somemachine1240",
  "Fqdn": "Somemachine1240.somedomain.com",
  "Status": 0,
  "OsType": "Linux",
  "OsVerNum": 1688863374311424,
  "OsVerTxt": "Ubuntu 14.1",
  "AgentVersion": 281483566841860,
  "Manufacturer": "DELL",
  "ChassisType": 7,
  "DeviceType": "Server",
  "CpuType": "Intel(R) Core i7 6700K 4.0GHz",
  "CpuArchitecture": "x64",
  "OsArchitecture": "x64",
  "RamMB": 32693,
  "SMBiosGuid": "ff2e3a5d-0395-423a-90bd-4d5c8f7b6a76",
  "TachyonGuid": "1a59b12e-a2cb-4ea4-926e-7d88c7e25a80",
  "LastBootUTC": "2019-02-04T0
	Use devices object inside the Tachyon connector instance and call GetDevices, while passing in an object containing filtering and sorting specification.
	
	Retrieving filtered, sorted and paginated devices Expand source
	
	var searchSettings = new Search
	{
	  Start = 1,
	  PageSize = 3,
	  Filter = new ExpressionObject
	  {
	  Attribute = "OsType",
	  Operator = "==",
	  Value = "Linux"
	  }
	}
	var devices = connector.Devices.GetDevices(searchSettings);
	
	devices object contains the same data as the JSON response on the left.

```
0:00:00Z",
            "LastConnUtc": "2019-03-07T0
8:14:35.93Z",
            "CreatedUtc": "2019-02-03T0
0:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-09
T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        },
        {
            "Id": 42,
            "Name": "Somemachine1241",
            "Fqdn": "Somemachine1241.som
edomain.com",
            "Status": 0,
            "OsType": "Linux",
            "OsVerNum": 168886337431142
4,
            "OsVerTxt": "Ubuntu 14.1",
            "AgentVersion": 281483566841
860,
            "Manufacturer": "DELL",
            "ChassisType": 7,
            "DeviceType": "Server",
            "CpuType": "Intel(R) Core i7
6700K 4.0GHz",
            "CpuArchitecture": "x64",
            "OsArchitecture": "x64",
            "RamMB": 32693,
            "SMBiosGuid": "837cdff5-a3bb
-4875-8158-da69751f607a",
            "TachyonGuid": "89044184-226
3-40e4-8fe8-83bb2215568e",
            "LastBootUTC": "2019-02-04T0
0:00:00Z",
            "LastConnUtc": "2019-03-07T0
```

```
8:14:35.93Z",
            "CreatedUtc": "2019-02-03T0
0:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-09
T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        },
        {
            "Id": 43,
            "Name": "Somemachine1242",
            "Fqdn": "Somemachine1242.som
edomain.com",
            "Status": 0,
            "OsType": "Linux",
            "OsVerNum": 168886337431142
4,
            "OsVerTxt": "Ubuntu 14.1",
            "AgentVersion": 281483566841
860,
            "Manufacturer": "DELL",
            "ChassisType": 7,
            "DeviceType": "Server",
            "CpuType": "Intel(R) Core i7
6700K 4.0GHz",
            "CpuArchitecture": "x64",
            "OsArchitecture": "x64",
            "RamMB": 32693,
            "SMBiosGuid": "f911c21f-c764
-46d2-8a0d-d07666937475",
            "TachyonGuid": "1436299e-77f
f-46a9-8b9c-665351ed6da3",
            "LastBootUTC": "2019-02-04T0
0:00:00Z",
            "LastConnUtc": "2019-03-07T0
8:14:35.93Z",
            "CreatedUtc": "2019-02-03T0
```

```
0:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-09
T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        }
    ]
}
```

2.5.4.2.2 Finding specific computers

A computer has two properties that on their own should be enough to identify it:

- Fully Qualified Domain Name (FQDN) - each FQDN in an organization should be unique
- Tachyon GUID - created by the Tachyon Agent to uniquely identify a computer.

You can use either FQDN or Tachyon GUID to locate a specific computer.

In edge cases, a computer might be replaced (for example, during an upgrade) and the new computer given the same FQDN . Although the FQDN will be identical, the Tachyon GUID will be different as it's created during the Tachyon Agent install on a new computer.

2.5.4.2.2.1 Finding a Device by its Fully Qualified Domain Name (FQDN)

Because machine's FQDN might contain characters that are not URI-friendly, FQDN sent to the Consumer API has to be Base64 encoded.

If I wanted to look for a computer whose FQDN is "Somemachine.somedomain.com", I would have to base64 encode it and use the encoded string as parameter in the API call.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20= will yield following response: **Response received from** https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20= Expand source `{ "Id": 1, "Name": "Somemachine", "Fqdn": "Somemachine.somedomain.com", "Status": 0, "OsType": "Windows", "OsVerNum": 1688863374311424, "OsVerTxt": "Microsoft Windows 7 Enterprise", "AgentVersion": 281483566841860, "Manufacturer": "Dell Inc.", "ChassisType": 7, "DeviceType": "Desktop", "CpuType": "Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz", "CpuArchitecture": "x64", "OsArchitecture": "x64", "RamMB": 32693, "SMBiosGuid": "55f144f2-c52b-4349-add4-1467dc34449e", "TachyonGuid": "eadad09d-1dd0-4f72-966a-f53d1302875f", "LastBootUTC": "2019-02-04T00:00:00Z", "LastConnUtc": "2019-03-07T08:14:35.93Z", "CreatedUtc": "2019-02-03T00:00:00Z", "VrPlatform": "", "TimeZone": 0, "CertType": "Domain", "CertExpiryUtc": "2019-10-09T04:53:24Z", "PushToken": null, "Model": "Precision T5600", "Domain": "Somedomain", "Tags": "||", "ConnectionState": [], "LocalIpAddress": null, "TimeZoneId": null, "SerialNumber": null, "Criticality": 1, "CoverageTags": {} }`	Here FQDN doesn't have to be Base64 encoded because the SDK performs the encoding internally. Use Devices object inside the Tachyon connector instance. **Retrieving device by its FQDN** Expand source `var device = connector.Devices.GetDeviceByFqdn("Somemachine.somedomain.com");` Device object contains the same data as the JSON response on the left.

2.5.4.2.2.2 Finding a Device by its Tachyon GUID

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD **Response received from** https://my.tachyon.server/Consumer/Devices/tachyonguid/1D886362-1D37-4BBB-9E39-CABA431593AD Expand source `{ "Id": 50, "Name": "Somemachine", "Fqdn": "Somemachine.somedomain.com", "Status": 0, "OsType": "Linux", "OsVerNum": 1688863374311424, "OsVerTxt": "Ubuntu 14.1", "AgentVersion": 281483566841860, "Manufacturer": "DELL", "ChassisType": 7, "DeviceType": "Server", "CpuType": "Intel(R) Core i7 6700K 4.0GHz", "CpuArchitecture": "x64", "OsArchitecture": "x64", "RamMB": 32693, "SMBiosGuid": "e474c20c-2aac-46ad-9bf7-0bda1c936e13", "TachyonGuid": "1d886362-1d37-4bbb-9e39-caba431593ad", "LastBootUTC": "2019-02-04T00:00:00Z", "LastConnUtc": "2019-03-07T08:14:35.93Z", "CreatedUtc": "2019-02-03T00:00:00Z", "VrPlatform": "", "TimeZone": 0, "CertType": "Domain", "CertExpiryUtc": "2019-10-09T04:53:24Z", "PushToken": null, "Model": "Poweredge", "Domain": "SomeDomain", "Tags": "||", "ConnectionState": [], "LocalIpAddress": null, "TimeZoneId": null, "SerialNumber": null, "Criticality": 1,`	Use devices object inside the Tachyon connector instance. **Retrieving device by its Tachyon GUID** Expand source `var device = connector.Devices.GetDeviceByGuid("1D8863621D374BBB9E39CABA431593AD");` device object contains the same data as the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "CoverageTags": {}
}
```

2.5.4.2.3 Finding management groups for computers

Tachyon supports organizing computers into management groups. A single computer can belong to many groups and at least belong to the All Devices group. All computers belong to the All Devices group by default and cannot be removed from it.

Management groups affect how instructions are targeted , so it's helpful to find out what groups computers belong to . You can do this by looking a computer up by its FQDN or Tachyon GUID as described in finding a specific computer.

2.5.4.2.3.1 By Fully Qualified Domain Name (FQDN)

Just as before, we have to use base64 encoded FQDN.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20=/ManagementGroups **Reponse received from** https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20=/ManagementGroups Expand source `[ { "Id": 1, "Name": "All Devices", "Description": "All devices are members of this ManagementGroup", "Expression": null, "Type": 0, "Count": -1, "UsableId": "global", "HashOfMembers": null, "CreatedTimestampUtc": "2019-03-06T11:48:53.58Z", "ModifiedTimestampUtc": "2019-03-06T11:48:53.58Z"`	Use Devices object inside the Tachyon connector instance. **Finding out which Management Groups device belong to using its FQDN** Expand source `var device = connector.Devices.GetManagementGroupsForDeviceFqdn("Somemachine.somedomain.com");` Device object contains the same data as the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

2.5.4.2.3.2 By Tachyon GUID

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD/ManagementGroups
 	
Response received from https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD/ManagementGroups Expand source
 	
`[ { "Id": 1, "Name": "All Devices", "Description": "All devices are members of this ManagementGroup", "Expression": null, "Type": 0, "Count": -1, "UsableId": "global", "HashOfMembers": null, "CreatedTimestampUtc": "2019-03-06T11:48:53.58Z", "ModifiedTimestampUtc": "2019-03-06T11:48:53.58Z" } ]`
	Use Devices object inside the Tachyon connector instance.
	
	Finding out which Management Groups device belong to using its Tachyon GUID Expand source
	
	`var guid = Guid.Parse("1D8863621D374BBB9E39CABA431593AD");`
	`var device = connector.Devices.GetManagementGroupsForDeviceTachyonGuid(guid);`
	
	Device object contains the same data as the JSON response on the left.

2.5.4.3 Find computers matching scope

Using scope allows you to limit which computers receive an instruction. Read defining the scope to find out more about how scope works, and narrowing the scope for examples.

Although using scope can seem like filtering, there are some differences, for example scope:

- Supports tags and management groups while regular filtering does not
- Does not support sorting or pagination so all results are returned in one go.

Some differences relate to the intended use of both features, for example:

- Filtering's design allows you to view the estate
- Scope's design allows you to narrow the target device audience of an instruction.

Scope expressions are standard Tachyon filter expressions. To find out more, read using scope and filter expressions and defining a filter.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Devices/scope Expand source
 	
`{ "Operator": "AND", "Operands": [{ "Attribute": "OsType", "Operator": "==", "Value": "Linux" }, { "Attribute": "model", "Operator": "==", "Value": "Cisco UCS" }] }`
 	
Returns this response:
 	
Return payload Expand source
 	
`[{ "Id": 49, "Name": "Somemachine1248", "Fqdn": "Somemachine1248.somedomain.com", "Status": 0, "OsType": "Linux", "OsVerNum": 1688863374311424, "OsVerTxt": "Ubuntu 14.1", "AgentVersion": 281483566841860, "Manufacturer": "DELL", "ChassisType": 7, "DeviceType": "Server", "CpuType": "Intel(R) Core i7 6700K 4.0GHz",`
	Use Device object inside the Tachyon connector instance.
	
	Retrieving devices matching scope Expand source
	
	`var expression = new ExpressionObject { Operator = "AND", Operands = new List<ExpressionObject> { new ExpressionObject { Attribute = "OsType", Operator = "==", Value = "Linux" }, new ExpressionObject { Attribute = "model", Operator = "==", Value = "Cisco UCS" } } };`
	
	`var statistics = connecto`

Direct Consumer API call	C# code using Consumer SDK library

<pre>"CpuArchitecture": "x64",
"OsArchitecture": "x64",
"RamMB": 32693,
"SMBiosGuid": "25a0acf0-dddd-4b0e-8d71-7ac5869be4b5",
"TachyonGuid": "5a3ea68c-76f7-4b81-8428-c0896755105c",
"LastBootUTC": "2019-02-04T00:00:00Z",
"LastConnUtc": "2019-03-07T08:14:35.93Z",
"CreatedUtc": "2019-02-03T00:00:00Z",
"VrPlatform": "",
"TimeZone": 0,
"CertType": "Domain",
"CertExpiryUtc": "2019-10-09T04:53:24Z",
"PushToken": null,
"Model": "Cisco UCS",
"Domain": "SomeDomain",
"Tags": "||",
"ConnectionState": [],
"LocalIpAddress": null,
"TimeZoneId": null,
"SerialNumber": null,
"Criticality": 1,
"CoverageTags": {}
},
{
"Id": 50,
"Name": "Somemachine1249",
"Fqdn": "Somemachine1249.somedomain.com",
"Status": 0,
"OsType": "Linux",
"OsVerNum": 1688863374311424,
"OsVerTxt": "Ubuntu 14.1",
"AgentVersion": 281483566841860,
"Manufacturer": "DELL",
"ChassisType": 7,
"DeviceType": "Server",
"CpuType": "Intel(R) Core i7 6700K 4.0GHz",
"CpuArchitecture": "x64",
"OsArchitecture": "x64",
"RamMB": 32693,
"SMBiosGuid": "e474c20c-2aac-46ad-9bf7-0bda1c936e13",
"TachyonGuid": "1d886362-1d37-4bbb-9e39-caba431593ad",
"LastBootUTC": "2019-02-04T00:00:00Z",</pre>
	<pre>r.Devices.GetDevicesMatchingScope(expression);</pre>
	Devices object contains the same data as the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library


```
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Cisco UCS",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    }
]
```


2.5.4.4 Find an approximate target for an instruction

When issuing instructions, think about your permissions, as they limit which management groups you can target.

An approximate target:

- Is calculated in the context of a specific instruction definition
- Accepts an optional scope expression
- Gives you an overview on how the instruction you're about to issue will affect your estate.

Returned information is aggregated by computer and operating system type. You cannot configure this aggregation.

The following example is using the "1E-Explorer-TachyonCore-InstalledOS" instruction definition. This instruction definition has an Id of 124 on my Tachyon installation, so if I would like to know what part of the estate would be affected if I sent this instruction, I should use this Id when making the reques.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Devices/approxtarget/124 without any payload returns this response: **Return payload** Expand source	Use Devices object inside the Tachyon connector instance. **Retrieving information about affected devices** Expand source

Direct Consumer API call	C# code using Consumer SDK library

<pre>{
    "TotalDevicesOnline": 0,
    "TotalDevicesOffline": 50,
    "ByDeviceType": [
        {
            "AggregateValue": "Desktop",
            "CountOffline": 40,
            "CountOnline": 0
        },
        {
            "AggregateValue": "Server",
            "CountOffline": 10,
            "CountOnline": 0
        }
    ],
    "ByOsType": [
        {
            "AggregateValue": "Linux",
            "CountOffline": 10,
            "CountOnline": 0
        },
        {
            "AggregateValue": "Windows",
            "CountOffline": 40,
            "CountOnline": 0
        }
    ]
}</pre>
	<pre>var statistics = connector.Devices.GetApproximateTarget(null, 124);</pre>
	Statistics object contains the same data as the JSON response on the left.

To check the effect the same instruction has on my estate when issued with a scope I should add the scope to the request's payload:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Devices/approxtarget/124 Expand source
 	
<pre>{
        "Operator": "AND",
        "Operands":[{
            "Attribute": "OsType",
            "Operator": "==",
            "Value": "Linux"
        },
        {</pre>
	Use Devices object inside the Tachyon connector instance.
	
	Retrieving information about affected devices, narrowed by scope Expand source
	
	<pre>var expression = new ExpressionObject
	{
	    Operator = "AND",
	    Operands = new List<ExpressionObject>
	    {</pre>

Direct Consumer API call	C# code using Consumer SDK library

```
            "Attribute": "model",
            "Operator": "==",
            "Value": "Cisco UCS"
        }]
}
```

Returns following response:

Return payload Expand source

```
{
    "TotalDevicesOnline": 0,
    "TotalDevicesOffline": 2,
    "ByDeviceType": [
        {
            "AggregateValue": "Server",
            "CountOffline": 2,
            "CountOnline": 0
        }
    ],
    "ByOsType": [
        {
            "AggregateValue": "Linux",
            "CountOffline": 2,
            "CountOnline": 0
        }
    ]
}
```

```
        new ExpressionObject
        {
            Attribute = "OsType",
            Operator = "==",
            Value = "Linux"
        },
        new ExpressionObject
        {
            Attribute = "model",
            Operator = "==",
            Value = "Cisco UCS"
        }
    }
};

var statistics = connector.Devices.G
etApproximateTarget(expression, 12
4);
```

Statistics object contains the same data as the JSON response on the left.

2.5.5 Handling errors returned by Tachyon

This section describes how the Tachyon Consumer API shows errors.

2.5.5.1 What does the Consumer API return when an error happens?

An HTTP call will result in a response message with a payload and HTTP status code. Usually, 4XX and 5XX codes represent errors, with 4XX range being errors made by the calling user (for example, incorrect payload or wrong URI) and 5XX errors on the server itself. The payload of the response message is a JSON array. Each element of that array is an Error object described in the next section.

2.5.5.2 The Error object

The Error object has 3 properties:

- ErrorCode - the internal Tachyon error code described in error code
- Message - a localized message displayed to a user

- Data - localization data, see localization.

Although these properties are always present, only Message is intended for end users.

2.5.5.2.1 Error code

Error code is designed to help support and development teams diagnose the error.

Because it's a string instead of a number it should be descriptive enough in most circumstances. The error code can be used by an end user to determine what the error is, although it's better to use the Message field as this will contain more details about why the error occurred and possible help about how to fix the error.

2.5.5.2.2 Localization

Localization is not currently supported. All errors are in English.

> **Note**
>
> To support Localization in the future a Data array is included, containing non-translatable parts of the error message. However, at this time, you can ignore this array.

2.5.5.3 Example error

Example error payload as returned by the Consumer API

```
[
    {
        "ErrorCode": "InstructionDefinition.CannotDeleteDueToPendingInstruction
s",
        "Message": "Instruction definition 1E-Explorer-TachyonCore-BiosDetails
cannot be deleted because there are pending instructions based on it.",
        "Data": [
            "1E-Explorer-TachyonCore-BiosDetails"
        ]
    }
]
```

2.5.5.4 Errors in Consumer API SDK

If you're using Consumer API .NET SDK, the errors are returned inside the ApiCallResponse class in the property called Errors, which is a collection. If the call was unsuccessful and the Success property is set to False, the Errors collection should contain the errors received. The Error class (contained by the Errors collection) has an additional field called ErrorType:

If ErrorType is equal to "Tachyon":

Means the error was returned by the Tachyon Consumer API and it will have the structure described above.

If ErrorType is equal to "Exception":

Means an exception was thrown inside the SDK either during the course of the HTTP call itself, or during subsequent model de-serialization. The exception's message, along with messages for its inner exceptions, will be in the Message property.

If ErrorType is equal to "Other":

Means that another type of error has occurred. This most likely means that although the HTTP call itself concluded successfully, the HTTP status code received indicated an error but the error could not be translated to the standard Error structure seen above. It might have been, for instance, an HTML page. This can happen if the address for Consumer API is incorrect and the SDK contacts an address that does not belong to Consumer API, or if the request is rejected by IIS without being processed by Consumer API at all.

2.5.5.5 Finding out more about the error

To find out more about the error, look in the Consumer API and Coordinator service log files. These log files are found on the server where the Tachyon master stack is installed, by default under %PROGRAMDATA%\1E\Tachyon

2.5.6 Issue an instruction, track its progress and retrieve responses

Issuing an instruction is the creation of a new instruction by Consumer API, as well as its workflow and response tables by the Coordinator. Issuing an instruction is done by sending an appropriate JSON to the API endpoint responsible for creating new instructions.

For this section you should:

- Know what instruction to issue - read about searching instructions in find the right instruction definition
- Have an issued instruction that does not require authentication or authorization and follows the default workflow.

The C# examples assume you're using Tachyon Consumer SDK and you already have a correctly instantiated instance of Tachyon connector class in an object called connector.

All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API. In the following examples this detail is left out, stating that the returned object contains the data. For example, when we say that XYZ object contains certain data, this means the ReceivedObject contains that data, since that's always true.

In most circumstances, the lifecycle of an Instruction is:

1. A user issuing the instruction, through Explorer or the Consumer API directly.
2. The Consumer API accepting the instruction, after which it is sent to Agents.
3. The Agents which execute the instruction, sending back any data it has produced or indicate a failure.

This data is made available to consumers (like Tachyon Explorer).

2.5.6.1 Issuing an instruction

Pick an Instruction Definition to base your new Instruction on. To issue that instruction, use a POST to the Instructions controller - sending a JSON object describing what the instruction should be.

The endpoint you should send the data to is https://my.tachyon.server/Consumer/Instructions , where my.tachyon.server is replaced with the address of your Tachyon instance.

2.5.6.1.1 Simple instruction

This example covers the "How are network adapters configured?" instruction definition. It does not require any parameters and it will be sent to all connected agents.

In this example, the instruction definition has the Id 1120, but we could also use its name, which is "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails". You cannot use both Id and Name because it may introduce ambiguity.

Direct Consumer API call	C# code using Consumer SDK library

If you wish to use the Id you can make a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source
 	
`{ "DefinitionId": 1120, "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 60, "KeepRaw": 1 }`
 	
If you wish to use the Name instead, you can make a POST request with this payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source
 	
`{ "DefinitionName": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails", "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 60, "KeepRaw": 1 }`
 	
In both cases the request returns this response:
 	
Return payload Expand source
 	
`{ "Id": 2004, "Sequence": 0, "Name": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails", "Description": "Get the configuration of the network adapters. Windows`
	Use the Instructions object inside the Tachyon connector instance.
	
	If you wish to use the instruction definition's Id:
	
	Issuing a new Instruction using instructions definition's Id Expand source
	
	`var newInstruction = connector.Instructions.SendInstruction(1120, null, 60, 120); if (newInstruction.Success) { var idToTrack = newInstruction.ReceivedObject.Id; // track instruction and retrieve responses } else { foreach (var error in newInstruction.Errors) { // display error } }`
	
	If you wish to use the instruction definition's name:
	
	Issuing a new Instruction using instructions definition's name Expand source
	
	`var newInstruction = connector.Instructions.SendInstruction("1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails", null, 60, 120); if (newInstruction.Success)`

Direct Consumer API call

```
Only.",
    "InstructionType": 0,
    "ReadablePayload": "How are netwo
rk adapters configured?",
    "Cmd": "SendAll",
    "Schema": [
        {
            "Name": "Description",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "DHCPEnabled",
            "Type": "bool",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseExpires
",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseObtaine
d",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPServer",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSDomain",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSHostName",
            "Type": "string",
            "Length": 32,
```

C# code using Consumer SDK library

```
{
   var idToTrack = newInstruction.Rece
ivedObject.Id;
   // track instruction and retrieve r
esponses
}
else
{
   foreach (var error in newInstructio
n.Errors)
   {
      // display error
   }
}
```

In either case, the newInstruction object will contain the same data you can see in the Json response on the left.

```
            "RenderAs": null
        },
        {
            "Name": "DNSServerSearchO
rder",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPAddress",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPSubnet",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "MACAddress",
            "Type": "string",
            "Length": 64,
            "RenderAs": null
        }
    ],
    "Aggregation": null,
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "CreatedTimestampUtc": "2018-04-1
8T13:29:25.09Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": null,
    "InstructionDefinitionId": 1120,
    "CreatedBy": "SomeDomain\\SomeUse
r",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
```

Direct Consumer API call | C# code using Consumer SDK library

```
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": null,
    "ResponseTemplateConfiguration":
null,
    "Workflow": "{\"StateMachine\":\"
State\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```

The data sent to the API includes:

Term	Definition
DefinitionId or **DefinitionName**	You can use either DefinitionId or DefinitionName, but not both as that can introduce ambiguity. DefinitionId should be the Id of the instruction definition your instruction should be based upon. DefinitionName should be the name of the instruction definition your instruction should be based upon.
InstructionTtlMinutes	Amount of time responses are gathered for. This TTL starts when the instruction goes live (is sent to agents), not when it's received by the API. This is because there might be a authentication and/or approval workflow to execute between the API receiving the instruction and it going live. Also, the instruction might never go live if it's rejected or stuck indefinitely in approval.
ResponseTtlMinutes	Amount of time the responses are available inside Tachyon after they've been gathered. When InstructionTtlMinutes elapses, ResponseTtlMinues starts.
KeepRaw	Is only used by aggregated instructions and defines if raw results should be kept next to aggregated ones.

Term	Definition
	If you choose not to keep raw results, only aggregated ones will be available, so you'll not be able to drill down into the details of each aggregate. Though not strictly necessary, this setting was included to make sure raw results are kept, as if they're not sent, this setting will default to "false" and raw results will not be kept.

DefinitionId (or DefinitionName), InstructionTtlMinutes and ResponseTtlMinutes form the minimum information to provide to the API when issuing the instruction.

The recieved response contains information about the instruction consisting of the data you've sent, like the parameters, scope, TTLs as well as information inherited from the Instruction Definition it was based on. The most important piece of information is the Id of the instruction (in this case its 2004), which will be used to track the instruction's progress and retrieve responses.

2.5.6.1.2 Adding parameters

Some instructions require data for the instruction to work. This data is supplied in the form of a collection of parameters, each one a key - value pair. The key is the parameter name and can be obtained from the Instruction Definition.

The Instruction Definition describes the parameters and the rules each value must comply to. These parameters are added into the appropriate locations in the instruction server side when it's issued.

This example uses the Echo instruction definition ("1E-Explorer-TachyonAgent-Echo"), which in my Tachyon installation has an id of 60, and the following definition:

Direct Consumer API call	C# code using Consumer SDK library
You can retrieve the instruction definition using its Id by making a GET request to **https://my.tachyon.server/Consumer/InstructionDefinitions/id/60**, or using its name by making a request to **https://my.tachyon.server/Consumer/InstructionDefinitions/Name/1E-Explorer-TachyonAgent-Echo**. Whichever method you choose the return payload will look something like this: **Response received from https://my.tachyon.server/Consumer/InstructionDefinitions/id/60** Expand source `{ "Id": 60, "Name": "1E-Explorer-TachyonAgent-Echo", "Description": "Send an echo request to devices",`	Use the InstructionDefinitions object inside the Tachyon connector instance. You can retrieve the definition by its Id: **Retrieving instruction definition details by Id** Expand source `var definition = connector.InstructionDefinitions.GetInstructionDefinition(60);` or by its name:

Direct Consumer API call	C# code using Consumer SDK library


```
    "InstructionSetId": 2,
    "InstructionSetName": "Everything",
    "InstructionType": 0,
    "ReadablePayload": "Echo message %msg%.",
    "Parameters": [
        {
            "Name": "msg",
            "Pattern": "%msg%",
            "DataType": "string",
            "ControlType": "freeText",
            "ControlMetadata": null,
            "Placeholder": "message to echo",
            "DefaultValue": null,
            "Validation": {
                "Regex": null,
                "MaxLength": "200",
                "AllowedValues": null,
                "NumValueRestrictions": null
            },
            "Value": null,
            "HintText": null,
            "Source": null
        }
    ],
    "Schema": [
        {
            "Name": "Message",
            "Type": "string",
            "Length": 256,
            "RenderAs": null
        }
    ],
    "Aggregation": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 120,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 1440,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
    "ResponseTemplateId": null,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": null,
    "Version": "7",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2018-03-08T15:22:11.2Z
```


Retrieving instruction definiti on details by Name Expand source

```
var definition = connect
or.InstructionDefinition
s.GetInstructionDefiniti
on("1E-Explorer-TachyonA
gent-Echo");
```

In either case, definition object will contain the same data you can see in the JSON response on the left

Direct Consumer API call	C# code using Consumer SDK library

```
",
    "NumberOfTimesExecuted": 6
}
```

As we can see the Parameters array has a single parameter in this instruction definition. This object defines a parameter that has to be provided when this instruction is issued.

Direct Consumer API call	C# code using Consumer SDK library

You can issue an instruction using the instruction definition's Id by making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source

```
{
    "DefinitionId": 60,
    "Parameters": [{
        "Name": "msg",
        "Value": "hello!"
    }],
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 120,
    "KeepRaw": 1
}
```

or using the instruction definition's name by making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source

```
{
    "DefinitionName": "1E-Explorer-TachyonAgent-Echo",
    "Parameters": [{
        "Name": "msg",
        "Value": "hello!"
    }],
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 120,
    "KeepRaw": 1
}
```

Use Instructions object inside the Tachyon connector instance.

Parameters are passed as a dictionary where key is the name of the parameter and value is the value you wish the parameter to have.

You can use instruction definition's Id:

Issuing a new Instruction with a parameter Expand source

```
var parameters = new Dictionary<string,string>{ {"msg", "hello!"} };
var newInstruction = connector.Instructions.SendInstruction(60, parameters, 60, 120);
if (newInstruction.Success)
{
   var idToTrack = newInstruction.ReceivedObject.Id;
   // track instruction and retrieve responses
}
else
{
   foreach (var error in newInstruction.Errors)
   {
      // display error
```

Direct Consumer API call	C# code using Consumer SDK library

Both calls will return the same response:

Return payload Expand source

```
{
    "Id": 2006,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonAgent-Echo",
    "Description": "Send an echo request to devi
ces",
    "InstructionType": 0,
    "ReadablePayload": "Echo message hello!.",
    "Cmd": "SendAll",
    "Schema": [
        {
            "Name": "Message",
            "Type": "string",
            "Length": 256,
            "RenderAs": null
        }
    ],
    "Aggregation": null,
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 120,
    "CreatedTimestampUtc": "2018-04-19T07:10:01.
567Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": null,
    "InstructionDefinitionId": 60,
    "CreatedBy": "SomeDomain\\SomeUser",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": "[{\"Name\":\"msg\",\"Patte
rn\":\"%msg%\",\"DataType\":null,\"ControlType\
":null,\"ControlMetadata\":null,\"Placeholder\":
null,\"DefaultValue\":null,\"Validation\":null,\
"Value\":\"hello!\",\"HintText\":null,\"Source\
```

```
    }
}
```

or its name:

Issuing a new Instruction with a parameter Expand source

```
var parameters = new Dictio
nary<string,string>{ {"msg
", "hello!"} };
var newInstruction = connec
tor.Instructions.SendInstru
ction("1E-Explorer-TachyonA
gent-Echo", parameters, 60,
120);
if (newInstruction.Success)
{
   var idToTrack = newInstr
uction.ReceivedObject.Id;
   // track instruction and
retrieve responses
}
else
{
   foreach (var error in ne
wInstruction.Errors)
   {
      // display error
   }
}
```

In either case, newInstruction object will contain the same data you can see in the Json response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
":null}]",
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": null,
    "ResponseTemplateConfiguration": null,
    "Workflow": "{\"StateMachine\":\"State\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```

In the response payload above, the parameters are now completed.

The Readable payload of the Instruction Definition changed from "Echo message %msg%." to "Echo message hello!." because the value of the "msg" parameter was "hello!"

If an instruction has more parameters there would be other entries in the array of parameters.

2.5.6.1.3 Narrowing the scope

If you do not want the instruction sent to all agents but to a subset, you can use the scope and describe the properties of the computers you want to target. To use the scope fill in the Scope field in the payload sent to the API.

You should read Defining the scope for detailed information about scope and expressions, including the fields you can use in the scope.

This example defines a scope sending the instruction to Windows 7 computers :

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source
 	
`{`
`"DefinitionId": 1120,`
`"InstructionTtlMinutes": 60,`
`"ResponseTtlMinutes": 60,`
`"Scope": {`
`"Operator": "AND",`
`"Operands": [{`
`"Attribute": "OsType",`
`"Operator": "==",`
`"Value": "Windows"`
`},`
	Use the Instructions object inside the Tachyon connector instance.
	
	Here you'll have to create a send model representing an instruction.
	
	Issuing a new Instruction with a scope Expand source
	
	`var instruction = new Tachyon.SDK.Consumer.Models.Send.Instruction`
	`{`
	`DefinitionId = 1120,`
	`InstructionTtlMinutes = 6`

Direct Consumer API call

```
        {
            "Attribute": "OsVerTxt",
            "Operator": "Like",
            "Value": "%7%"
        }]
    }, "KeepRaw": 1
}
```

Returns the following response:

Return payload Expand source

```
{
    "Id": 2008,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonCore-NetworkAd
apterConfigurationDetails",
    "Description": "Get the configuration of t
he network adapters. Windows Only.",
    "InstructionType": 0,
    "ReadablePayload": "How are network adapte
rs configured?",
    "Cmd": "SendAll",
    "Schema": [
        {
            "Name": "Description",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "DHCPEnabled",
            "Type": "bool",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseExpires",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseObtained",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
```

C# code using Consumer SDK library

```
0,
   ResponseTtlMinutes = 120,
   KeepRaw = true,
   Scope = new ExpressionObje
ct
   {
      Operator = "AND",
      Operands = new List<Exp
ressionObject>
      {
         new ExpressionObject
         {
            Attribute = "OsTy
pe",
            Operator = "==",
            Value = "Windows"
         },
         new ExpressionObject
         {
            Attribute = "OsVe
rTxt",
            Operator = "Like
",
            Value = "%7%"
         }
      }
   }
};

var newInstruction = connecto
r.Instructions.SendInstructio
n(instruction);
if (newInstruction.Success)
{
   var idToTrack = newInstruc
tion.ReceivedObject.Id;
   // track instruction and r
etrieve responses
}
else
{
   foreach (var error in newI
nstruction.Errors)
   {
      // display error
   }
}
```

Direct Consumer API call	C# code using Consumer SDK library

{
"Name": "DHCPServer",
"Type": "string",
"Length": 32,
"RenderAs": null
},
{
"Name": "DNSDomain",
"Type": "string",
"Length": 32,
"RenderAs": null
},
{
"Name": "DNSHostName",
"Type": "string",
"Length": 32,
"RenderAs": null
},
{
"Name": "DNSServerSearchOrder",
"Type": "string",
"Length": 65535,
"RenderAs": null
},
{
"Name": "IPAddress",
"Type": "string",
"Length": 65535,
"RenderAs": null
},
{
"Name": "IPSubnet",
"Type": "string",
"Length": 65535,
"RenderAs": null
},
{
"Name": "MACAddress",
"Type": "string",
"Length": 64,
"RenderAs": null
}
],
"Aggregation": null,
"KeepRaw": true,
"Scope": {
"Operator": "AND",
"Operands": [
	newInstruction object will contain the same data you can see in the JSON response on the left.

```
            {
                "Attribute": "OsType",
                "Operator": "==",
                "Value": "Windows",
                "DataType": "String"
            },
            {
                "Attribute": "OsVerTxt",
                "Operator": "Like",
                "Value": "%7%",
                "DataType": "String"
            }
        ]
    },
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "CreatedTimestampUtc": "2018-04-19T10:02:2
7.523Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": null,
    "InstructionDefinitionId": 1120,
    "CreatedBy": "SomeDomain\\SomeUser",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": null,
    "ResponseTemplateConfiguration": null,
    "Workflow": "{\"StateMachine\":\"State\"}
",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```

2.5.6.1.4 Targeting specific devices

You can also send instructions to a specific list of computers. To do this, you'll need to call a different API endpoint.

The https://my.tachyon.server/Consumer/Instructions/Targeted endpoint takes a payload almost identical to one used previously, when we were using https://my.tachyon.server/Consumer/Instructions, but contains an additional field called 'Devices'. The Devices should be a list of FQDNs of the computers to send the instruction to.

All supplied FQDNs have to be valid and known to Tachyon at the time the instruction's issued.

Depending on the permissions of the user making the API call, the list of FQDNs might be filtered to exclude devices that user doesn't have permissions to, based on Management groups associated with their permissions. If it turns out that the caller doesn't have permission to any of the devices whose FQDNs they provided (so when the list is filtered down to zero elements), an error will be retrurned.

Direct Consumer API call	C# code using Consumer SDK library

Like with a regular instruction, you can issue a targeted instruction using instruction definition's Id by making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Instructions/Targeted Expand source
 	
`{ "DefinitionId": 1120, "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 60, "Devices": ["Device.SomeDomain.local"], "KeepRaw": 1 }`
 	
or by using instruction definition's name by making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Instructions/Targeted Expand source
 	
`{ "DefinitionName": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails", "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 60, "Devices": ["Device.SomeDomain.local"],`
	Use Instructions object inside the Tachyon connector instance.
	
	To issue a targeted instruction using instruction definition's Id:
	
	Issuing a new targeted Instruction using instruction definition's Id Expand source
	
	`var newInstruction = connector.Instructions.SendTargetedInstruction(1120, null, 60, 60, new List<string> { "Device.SomeDomain.local" }); if (newInstruction.Success) { var idToTrack = newInstruction.ReceivedObject.Id; // track instruction and retrieve responses } else { foreach (var error in newInstruction.Errors) { // display error } }`
	
	To issue it using instruction definition's name:

Direct Consumer API call

C# code using Consumer SDK library

```
    "KeepRaw": 1
}
```

Returns this response:

Return payload Expand source

```
{
    "Id": 2010,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonCore-N
etworkAdapterConfigurationDetails",
    "Description": "Get the configurat
ion of the network adapters. Windows O
nly.",
    "InstructionType": 0,
    "ReadablePayload": "How are networ
k adapters configured?",
    "Cmd": "SendList",
    "Schema": [
        {
            "Name": "Description",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "DHCPEnabled",
            "Type": "bool",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseExpires
",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseObtained
",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPServer",
```

Issuing a new targeted Instruction using instr uction definition's name Expand source

```
var newInstruction = connector.Instruc
tions.SendTargetedInstruction("1E-Expl
orer-TachyonCore-NetworkAdapterConfigu
rationDetails", null, 60, 60, new List
<string> { "Device.SomeDomain.local
" });
if (newInstruction.Success)
{
   var idToTrack = newInstruction.Rece
ivedObject.Id;
   // track instruction and retrieve r
esponses
}
else
{
   foreach (var error in newInstructio
n.Errors)
   {
      // display error
   }
}
```

In either case, the newInstruction object will contain the same data you can see in the JSON response on the left.

```
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSDomain",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSHostName",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSServerSearchOr
der",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPAddress",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPSubnet",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "MACAddress",
            "Type": "string",
            "Length": 64,
            "RenderAs": null
        }
    ],
    "Aggregation": null,
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "CreatedTimestampUtc": "2018-04-19
```

Direct Consumer API call	C# code using Consumer SDK library


```
T11:07:19.4Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": null,
    "InstructionDefinitionId": 1120,
    "CreatedBy": "SomeDomain\\SomeUser
",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": null,
    "ResponseTemplateConfiguration": n
ull,
    "Workflow": "{\"StateMachine\":\"S
tate\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```


The response does not contain the list of devices because it's kept in a form consumable by the Switch and not as FQDNs.

Converting this internal format back to FQDN list is expensive. If you want that list, you'll have to explicitly ask for it using a different endpoint: https://my.tachyon.server/Consumer/Instructions/{instructionId}/targetlist

Where instructionId is the Id of the instruction whose targets you want. The return payload will be a list of FQDNs.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Instructions/2010/targetlist returns this response:	Use Instructions object inside the Tachyon connector instance. **Getting the list of target devices for given instruction** Expand source

Direct Consumer API call	C# code using Consumer SDK library

Response received from https://my.tachyon.server/Consumer/Instructions/2010/targetlist Expand source
 	
`[`
`  "1EUKDEVWKS1204.1e.local"`
`]`
	`var targets = connector.Instructions.GetTargetsByInstructionId(2010);`

2.5.6.1.5 Following up on an instruction

Follow up instructions are a way of chaining separate instructions so they're executing together and appear as if they're one instruction. Each instruction in the chain forms a part of the instruction by adding more code to execute.

To issue a follow up instruction you can use the APIs used in the examples and add "ParentInstructionId" fiield to the payload. The value of "ParentInstructionId" should be an Instruction Id.

Direct Consumer API call	C# code using Consumer SDK library

You can issue a follow up instruction using instruction definition's Id by making a POST request with following payload:
Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source
 	
`{`
`  "DefinitionId": 1120,`
`  "InstructionTtlMinutes": 60,`
`  "ResponseTtlMinutes": 60,`
`  "KeepRaw": 1,`
`  "ParentInstructionId": 2012`
`}`
 	
You can also use the instruction definition's name by making a POST request with following payload
 	
Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source
 	
`{`
`  "DefinitionName": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails",`
`  "InstructionTtlMinutes": 60,`
`  "ResponseTtlMinutes": 60,`
`  "KeepRaw": 1,`
	Use the Instructions object inside the Tachyon connector instance.
	
	Here you'll have to create a send model representing an instruction. You can either use the DefinitionId or DefinitionName field in Tachyon.SDK.Consumer.Models.Send.Instruction, but you cannot use both.
	
	Here's how you'd do it using instruction definition Id:
	
	Issuing a follow-up instruction using instruction definition's Id Expand source
	
	`var instruction = new Tachyon.SDK.Consumer.Models.Send.Instruction`
	`{`
	`  DefinitionId = 1120,`
	`  InstructionTtlMinutes = 60,`
	`  ResponseTtlMinutes = 120,`
	`  KeepRaw = true,`
	`  ParentInstructionId = 2012`
	`};`
	`var newInstruction = connector.Instructions.SendTargetedInstruction`

Direct Consumer API call

C# code using Consumer SDK library

```
    "ParentInstructionId": 2012
}
```

Returns this response: **Return payload** Expand source

```
{
    "Id": 2013,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonCore-Netw
orkAdapterConfigurationDetails",
    "Description": "Get the configuration
of the network adapters. Windows Only.",
    "InstructionType": 0,
    "ReadablePayload": "How are network a
dapters configured?",
    "Cmd": "SendList",
    "Schema": [
        {
            "Name": "Description",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "DHCPEnabled",
            "Type": "bool",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseExpires",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPLeaseObtained",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "DHCPServer",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
```

```
(instruction);
if (newInstruction.Success)
{
   var idToTrack = newInstruction.
ReceivedObject.Id;
   // track instruction and retrie
ve responses
}
else
{
   foreach (var error in newInstru
ction.Errors)
   {
      // display error
   }
}
```

and here's how you'd do it using the instruction definition's name:

Issuing a follow-up instruction using instruction definitio n's name Expand source

```
var instruction = new Tachyon.SDK.
Consumer.Models.Send.Instruction
{
   DefinitionName = "1E-Explorer-T
achyonCore-NetworkAdapterConfigura
tionDetails",
   InstructionTtlMinutes = 60,
   ResponseTtlMinutes = 120,
   KeepRaw = true,
   ParentInstructionId = 2012
};
var newInstruction = connector.Ins
tructions.SendTargetedInstruction
(instruction);
if (newInstruction.Success)
{
   var idToTrack = newInstruction.
ReceivedObject.Id;
   // track instruction and retrie
ve responses
}
else
{
   foreach (var error in newInstru
```

Direct Consumer API call

C# code using Consumer SDK library

```
        {
            "Name": "DNSDomain",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSHostName",
            "Type": "string",
            "Length": 32,
            "RenderAs": null
        },
        {
            "Name": "DNSServerSearchOrder
",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPAddress",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "IPSubnet",
            "Type": "string",
            "Length": 65535,
            "RenderAs": null
        },
        {
            "Name": "MACAddress",
            "Type": "string",
            "Length": 64,
            "RenderAs": null
        }
    ],
    "Aggregation": null,
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 60,
    "ResponseTtlMinutes": 60,
    "CreatedTimestampUtc": "2018-04-19T1
4:16:40.127Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
```

```
ction.Errors)
   {
      // display error
   }
}
```

In either case, the newInstruction object will contain the same data you can see in the JSON response on the left.

```
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": 2012,
    "InstructionDefinitionId": 1120,
    "CreatedBy": "SomeDomain\\SomeUser",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": null,
    "ResponseTemplateConfiguration": nul
l,
    "Workflow": "{\"StateMachine\":\"Stat
e\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```


Note

Only questions can be followed up on. Actions can be issued as a follow-up but cannot be a parent instruction (you cannot follow-up on them). This is because they could modify the endpoint, and if an action was already executed, issuing a follow-up based on it would perform the action again before performing the follow-up instruction. This is because of instruction chaining.

2.5.6.1.6 Adding results filters

You can use results filters if you're interested in responses with a specific value.

For example, you might only be interested in the configuration of network adapters for computers belonging to a specific subnet. Because subnet is part of the Instruction's schema, you can add a results filter with the subnet to an Instruction. Only agents with responses matching the filter will send a reply.

If you're asking a follow up instruction, you can add a previous results filter, based on columns returned by the parent instruction. Read results filter for an instruction for more information.

This section focuses on using filters from a coding perspective.

Results filter and previous results filter are both expressions and should be placed in the "ResultsFilter" and 'PreviousResultsFilter" fields in the payload you send to the API when issuing an instruction. Read more in using scope and filter expressions.

In this example, the "1E-Explorer-TachyonCore-AllInstalledSoftware" instruction is issued as a follow up to the instruction issued in the previous example.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Instructions Expand source

```
{
    "ParentInstructionId": 2013,
    "DefinitionId": 1051,
    "InstructionTtlMinutes": 10,
    "ResponseTtlMinutes": 30,
    "KeepRaw": 1,
    "ResultsFilter": {
        "Attribute": "Publisher",
        "Operator": "Like",
        "Value": "%Microsoft%"
    },
    "PreviousResultsFilter": {
        "Attribute": "DHCPServer",
        "Operator": "==",
        "Value": "192.168.2.5"
    }
}
```

Returns this response:

Return payload Expand source

```
{
    "Id": 2015,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonCore-AllInstalledSoftware",
    "Description": "Returns all installed software.",
    "InstructionType": 0,
    "ReadablePayload": "What software is installed?",
    "Cmd": "SendList",
    "Schema": [{
        "Name": "Product",
        "Type": "string",
        "Length": 512,
        "RenderAs": null
    },
    {
```

Use the Instructions object inside the Tachyon connector instance.

Issuing a follow-up instruction with result filter and previous results filter Expand source

```
var instruction = new Tachyon.SDK.Consumer.Models.Send.Instruction
{
   DefinitionId = 1051,
   InstructionTtlMinutes = 10,
   ResponseTtlMinutes = 30,
   KeepRaw = true,
   ParentInstructionId = 2013,
   ResultsFilter = new ExpressionObject
   {
      Operator = "Like",
      Attribute = "Publisher",
      Value = "Microsoft"
   },
   PreviousResultsFilter = new ExpressionObject
   {
      Operator = "==",
      Attribute = "DHCPServer",
      Value = "192.168.2.5"
   }
};
var newInstruction = connector.Instructions.SendTargetedInstruction(instruction);
if (newInstruction.Success)
{
   var idToTrack = newInstruction.ReceivedObject.Id;
   // track instruction and retrieve responses
}
else
{
   foreach (var error in newInstruction.Errors)
   {
```

Direct Consumer API call

```
        "Name": "Publisher",
        "Type": "string",
        "Length": 512,
        "RenderAs": null
    },
    {
        "Name": "Version",
        "Type": "string",
        "Length": 128,
        "RenderAs": null
    },
    {
        "Name": "InstallDate",
        "Type": "datetime",
        "Length": 0,
        "RenderAs": null
    },
    {
        "Name": "Architecture",
        "Type": "string",
        "Length": 20,
        "RenderAs": null
    }],
    "Aggregation": {
        "Schema": [{
            "Name": "Publisher",
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Product",
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Version",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "Count",
            "Type": "int64",
            "Length": 0,
            "RenderAs": null
        }],
```

C# code using Consumer SDK library

```
        // display error
    }
}
```

The newInstruction object will contain the same data you can see in the JSON response on the left.

```
        "GroupBy": "Publisher,Product,V
ersion",
        "Operations": [{
            "Name": "Count",
            "Type": "count"
        }]
    },
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 10,
    "ResponseTtlMinutes": 30,
    "CreatedTimestampUtc": "2018-04-20T
08:03:43.893Z",
    "SentTimestampUtc": null,
    "Status": 0,
    "WorkflowState": 0,
    "StatusTimestampUtc": null,
    "Export": false,
    "ExportLocation": "",
    "ParentInstructionId": 2013,
    "InstructionDefinitionId": 1051,
    "CreatedBy": "SomeDomain\\SomeUser
",
    "ResultsFilter": {
        "Attribute": "Publisher",
        "Operator": "Like",
        "Value": "%Microsoft%",
        "DataType": "string"
    },
    "PreviousResultsFilter": {
        "Attribute": "DHCPServer",
        "Operator": "==",
        "Value": "192.168.2.5",
        "DataType": "string"
    },
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": 1,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [{
            "Id": "mainchart",
            "Title": "Installed softwar
e - 5 most common application from 5 mo
```

```
st common publishers",
            "Type": "Bar",
            "X": "Product",
            "Y": "Count",
            "Z": "Publisher",
            "PostProcessor": "processin
gFunction",
            "Size": 1,
            "Row": 1
        }],
        "PostProcessors": [{
            "Name": "processingFunction
",
            "Function": "ProcessMultiSe
ries('Product', 'Count', 'Publisher', '
5', '5', 'false')"
        }]
    },
    "Workflow": "{\"StateMachine\":\"St
ate\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```

2.5.6.2 Tracking progress

Once the instruction is issued it becomes a part of a workflow. This workflow can include authentication and authorization steps and includes gathering responses, keeping them for a specific amount of time, optionally exporting them and performing a clean-up after the data gathered expires or is deemed no longer needed (for example, instruction was cancelled, see below).

Instructions can have one of the following statuses:

Status	Numerical representation	Description
Created	0	Instruction has just been created. This is a transitional state and the instruction will imminently move to 'InApproval', 'Authenticating' or 'Sent' state, depending on the workflow used by the instruction.

Status	Numerical representation	Description
InApproval	1	Instruction is awaiting approval. It will remain in this state until it is approved, rejected or cancelled.
Rejected	2	Instruction has been rejected at approval stage.
Approved	3	Instruction has been approved. This is a transitional state and instruction will imminently move to 'Sent' state.
Sent	4	Instruction has been sent. This is a transitional state and instruction will imminently move to 'InProgress' state.
InProgress	5	Instruction is live and responses are being gathered. It will remain in this state until Instruction TTL runs out or instruction is cancelled.
Complete	6	Instruction has finished. Responses are no longer being gathered, but are available for viewing and instruction will not be sent to any more agents. Instruction will remain in this state until Responses TTL runs out.
Expired	7	Instruction has expired. Responses are no longer available.
Cancelling	8	Instruction has been cancelled but any results gathered were kept. Responses are no longer being gathered, are available for viewing and instruction will not be sent to any more agents. Instruction will remain in this state until Responses TTL runs out.
Cancelled	9	Instruction has been cancelled and any results gathered were deleted. Responses are no longer being gathered, are not available and instruction will not be sent to any more agents.
Failed	10	Instruction has failed.
Authenticating	12	Instruction is awaiting authentication using two-factor authentication mechanism. It remains in this state until it's successfully authenticated, user(s) fail authentication 3 times or it is cancelled.

When the Status of the instruction becomes "InProgress" the instruction is 'live' and is being sent to agents for processing. While the instruction remains in this state, responses are gathered and new agents coming on-line receive the instruction for processing, providing they meet scope or target list criteria, if those were defined for the instruction.

Once the instruction TTL elapses, the instruction is complete and changes its status to "Complete". At this moment responses TTL timer starts ticking and once that elapses the status will change to "Expired" and all responses gathered will be deleted from the system. If when issuing an instruction an export option was set to true, responses are exported automatically before they're deleted.

You can track the Instruction's progress using its Id and two separate APIs, Instructions and Instruction Statistics:

Instructions API - allows you to retrieve details about the instruction, including its Status, to find out where in its life cycle the instruction is.

Instruction Statistics API - allows you to track progress of responses coming in from the agents, best used while instruction is in "InProgress".

2.5.6.2.1 Checking Instruction status

Calling https://my.tachyon.server/Consumer/Instructions/{id} returns current information about given instruction, including its status.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Instructions/2015 returns this response:	Use Instructions object inside the Tachyon connector instance.
Return payload Expand source	Expand source
`{ "Id": 2015, "Sequence": 24, "Name": "1E-Explorer-TachyonCore-AllInstalledSoftware", "Description": "Returns all installed software.", "InstructionType": 0, "ReadablePayload": "What software is installed?", "Cmd": "SendList", "Schema": [{ "Name": "Product", "Type": "string", "Length": 512, "RenderAs": null }, { "Name": "Publisher", "Type": "string",`	`var instruction = connector.Instructions.GetInstruction(2015);` object will contain the same data you can see in the JSON response on the left.

```
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Version",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "InstallDate",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "Architecture",
            "Type": "string",
            "Length": 20,
            "RenderAs": null
        }
    ],
    "Aggregation": {
        "Schema": [
            {
                "Name": "Publisher",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Product",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Version",
                "Type": "string",
                "Length": 128,
                "RenderAs": null
            },
            {
                "Name": "Count",
                "Type": "int64",
                "Length": 0,
                "RenderAs": null
```

```
            }
        ],
        "GroupBy": "Publisher,Product,Version",
        "Operations": [
            {
                "Name": "Count",
                "Type": "count"
            }
        ]
    },
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 10,
    "ResponseTtlMinutes": 30,
    "CreatedTimestampUtc": "2018-04-20T08:03:43.893Z",
    "SentTimestampUtc": "2018-04-20T08:03:44.333Z",
    "Status": 5,
    "WorkflowState": 4,
    "StatusTimestampUtc": "2018-04-20T08:43:45.417Z",
    "Export": false,
    "ExportLocation": "",
    "ParentInstructionId": 2014,
    "InstructionDefinitionId": 1051,
    "CreatedBy": "SomeDomain\\SomeUser",
    "ResultsFilter": {
        "Attribute": "Publisher",
        "Operator": "Like",
        "Value": "%Microsoft%",
        "DataType": "string"
    },
    "PreviousResultsFilter": {
        "Attribute": "DHCPServer",
        "Operator": "==",
        "Value": "192.168.2.5",
        "DataType": "string"
    },
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": 1,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [
            {
```

Direct Consumer API call	C# code using Consumer SDK library


```
                "Id": "mainchart",
                "Title": "Installed software - 5 most co
mmon application from 5 most common publishers",
                "Type": "Bar",
                "X": "Product",
                "Y": "Count",
                "Z": "Publisher",
                "PostProcessor": "processingFunction",
                "Size": 1,
                "Row": 1
            }
        ],
        "PostProcessors": [
            {
                "Name": "processingFunction",
                "Function": "ProcessMultiSeries('Product
', 'Count', 'Publisher', '5', '5', 'false')"
            }
        ]
    },
    "Workflow": "{\"StateMachine\":\"State\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
    "ActionReason": null
}
```


The Status of the instruction is 5. Looking at the table above this means "InProgress". Because the instruction's in progress, you can looking at its statistics and retrieve responses.

2.5.6.2.2 Checking instruction statistics

You can get statistics from https://my.tachyon.server/Consumer/InstructionStatistics API. You can also provide an Id of the instruction you're interested in to get its statistics based.

The code below gets statistics for the issued instruction when using the results filters.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/InstructionStatistics/2015 returns this response:	Use InstructionStatistics object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

GET to https://my.tachyon.server/Consumer/InstructionStatistics/2015 Expand source
 	
{
"Id": 2015,
"EstimatedCount": 1,
"ReceivedCount": 1,
"SentCount": 1,
"OutstandingResponsesCount": 0,
"AverageExecTime": "00:00:03.3950000",
"Runningtime": "00:10:00",
"TotalBytesSent": 538,
"TotalBytesReceived": 16942,
"AverageBytesReceived": 16942,
"TotalRowInserts": 1614,
"TotalRowProcessed": 1662,
"TotalSuccessRespondents": 1,
"TotalSuccessNoDataRespondents": 0,
"TotalErrorRespondents": 0,
"TotalNotImplementedRespondents": 0,
"TotalResponseTooLargeRespondents": 0,
"TotalSubscribedEventsRespondents": 0,
"InstructionId": 2015,
"InstructionDefinitionId": 1051,
"Sequence": 24
}
	Retrieving statistics for an Instruction Expand source
	
	`var stats = connector.InstructionStatistics.Get(2015);`
	
	stats object will contain the same data you can see in the JSON response on the left.

Returned fields:

Property	Description
EstimatedCount	Estimates count of devices the instruction would be sent to.
ReceivedCount	Exact count of devices that responded.
SentCount	Exact count of devices the instruction was sent to.
OutstandingResponsesCount	Exact count of devices that are yet to respond.
AverageExecTime	Average time Agent took to process the instruction.
Runningtime	How long this instruction has been 'live' for.

Property	Description
TotalBytesSent	Total number of bytes sent.
TotalBytesReceived	Total number of bytes received.
AverageBytesReceived	Average number of bytes received from each device.
TotalRowInserts	Total number of rows inserted into the database.
TotalRowProcessed	Total number of rows processed.
TotalSuccessRespondents	Total number of devices that reported "success".
TotalSuccessNoDataRespondents	Total number of devices that reported they've successfully processed the instruction but have no data to send back.
TotalErrorRespondents	Total number of devices that reported error.
TotalNotImplementedRespondents	Total number of devices that reported they cannot execute the instruction because it contains functionality not implemented on their operating system.
TotalResponseTooLargeRespondents	Total number of devices that reported they couldn't send a response because it was too large
TotalSubscribedEventsRespondents	Tachyon versions to 4.1 inclusive. Total number of devices that reported they have subscribed to the event. Only valid for event type instructions.
InstructionId	Id of the Instruction these statistics belong to.
InstructionDefinitionId	Id of the Instruction Definition the Instruction to whom these statistics belong to was based on
Sequence	Sequence number of the Instruction. This may be different from the Id because instructions can be sent to agents in a different order to the one they were issued in. This depends on the need to authorize or authenticate instructions.

To track the progress of the instruction, look at the Total[...]Respondents properties and compare the values with previous statistics. If the numbers have increased, there are new responses available.

2.5.6.3 Retrieving responses

Responses become available once the instruction has gone "live", so when its Status has changed to "InProgress". When you ask for responses, you'll receive the ones already stored in the system, but until the instruction has finished or been cancelled, there might be more responses coming in. Because some instructions can take more than a few seconds to execute, depending on how complex they are and what they are doing, you might not see responses immediately, but after a short delay. You should also take into account possible delays in instruction workflow, such as Tachon waiting for the issuing user to authenticate when issuing an instruction or waiting for instruction (usually action) to be approved. In general, if you're issuing an action, you should not expect it to immediatelly go 'live' (unless it uses a custom workflow which bypases authentication and/or approval) and instead you should periodicaly check if an instruction has gone 'live' i.e. its status has changed to 'InProgress'. It makes sense to start retireving responses only once that happened. An instruction that a user fails to authenticate or that is rejected during approval process will not go live and won't have any responses. If an instruction is forgotten, it can potentially be "stuck" in approval forever.

Once the instruction is live, you can start retrieving responses. If you wish to use polling mechanism to retrieve responses, there are two ways you can do this:

- Calling the Responses API to simply request responses, if there's nothing new, you'll receive an empty array of responses.
- Checking Instruction Statistics and examining the "TotalSuccessRespondents" property to see if it's increased, this would mean new agents have responded. In this case you should remember that even if this statistic doesn't change, there still might be responses you haven't fetched because they didn't fit on the last page you requested.

Using the checking Instruction Statistics option, gives you information about all types of responses, not just successful ones. If you poll for responses you will only receive successful responses as there's another API that needs to be called to retrieve other kinds of responses (see "Other responses" below). Checking the statistics helps you make a decision on which responses to retrieve and when to retrieve them.

You can aggregate instructions where by default only aggregated data is kept. To also have access to the raw data used to make the aggregates, you need to set the "KeepRaw" flag to true when issuing an instruction. If this flag is set to false, retrieving raw responses will not return an error. Instead, it will always return an empty array. Asking for aggregated responses for an instruction that's not aggregated results in an error.

Once the instruction ttl has elapsed, no new responses will be received from agents, but it might also happen that an instruction is cancelled before its ttl elapses. In this case, depending on whether the cancellation request specified that responses should be kept or not, responses might not be available after the cancellation takes effect.

When you are pooliing for responses you should be mindful of changes in the status of the instruction. If the instruction is allowed to run its course, the status will change from InProgress (5) to Complete (6). In this case, pooling for the duration od instruction ttl is fine. If the instruction is cancelled you should stop pooling when the status changes. If the status changed to Cancelling (8) then you can still retrieve responses. If the status

changed to Cancelled (9) then you won't be able to retrieve responses because they have been removed when the instruction was cancelled.

You can get successful responses from:

- https://my.tachyon.server/Consumer/Responses/{instructionId}/ - for raw
- https://my.tachyon.server/Consumer/Responses/{instructionId}/Aggregate - for aggregated.

Other responses are available from:

- https://my.tachyon.server/Consumer/ResponseErrors/{instructionId} - for raw
- https://my.tachyon.server/Consumer/ResponseErrors/{instructionId}/Aggregate - for aggregated.

Those endpoints require similar payloads and return similar data. The main difference between aggregated and non aggregated is that aggregation does not support pagination.

2.5.6.3.1 Pagination

The above paragraph focused on pooling in the context of establishing if new responses are available, but that is just one aspect of the issue.

You should also be mindful of the number of responses you retrieve each time you ask for them. If your page size is small, you might still have responses to retrieve even though no new responses have been received.

Let's look at an example estate with 5 agents. An instruction is issued to all agents and each of them returns 20 rows. This means you have 100 responses. If you ask for 25 responses, you will receive 25 rows and even though no new responses are received from agents, you still have 75 responses to be retrieved from the API.

Agents can return many, even more than a hundred, rows in response to a single instruction, like an instruction that returns all installed software. TotalSuccessRespondents has the count of agents that have responded and not how many rows have been returned.

In an example at the end of this page you will see TotalSuccessRespondents used as page size, but that assumes that only 1 row is returned by the agent.

2.5.6.3.2 How can I tell if I'm "done" with responses?

This is a very common question and here we'll try to answer the it in its strictest meaning - "When can I be sure there will be no more responses to retrieve?". I many cases, this isn't quite what you might be after, as you can decide you don't need any more responses after a certain percentage of agents have sent a response, or a particular agent has replied. We will not delve into these cases, since they represent particular problems and it should be up you, the user, to decide when you think you are "done" with responses. Instead, we will focus on determining when we've retrieved all possible responses for an instruction.

In general, while the instruction is 'live', so its status is InProgress (5) and it is within its ttl, you should assume that more responses can appear at any time. Even if the statistics report that you've sent the instruction to 10 devices and 10 have replied, there might be 11th coming online just before ttl expires and it will process the instruction and send a response.

The only exception to this rule is if you're sending a targeted instruction (using FQDNs) and you can see that all targeted devices have responded. Then, because you specified exactly who should respond to the instruction

and they all have responded, you can assume you're "done", even if instruction is still InProgress (in this instance it might be worth cancelling the instruction because you already have all the possible results).

If the instruction is no longer live then no new responses will be received because the instruction will not be sent to any new agents. By this stage aggregated responses would reach their final values. For raw responses, you should pick a page size that is appropriate for your needs and keep retrieving pages of responses up until the point where you receive fewer responses than the size of the page you defined. This would mean that because there weren't enough responses to fill the page, and no new responses will be coming in because the instruction is no longer live, you have retrieved all available responses. You can also retrieve everything in one go by specifying "0;0" start marker and maximum value allowed for an integer as page size but please remember that this might return a lot of data and pagination is usually preferred method of retrieving raw responses.

2.5.6.3.3 How can I tell if my instruction is aggregated or not?

Look at the "Aggregation" property of either Instruction or Instruction Definition. If it's null, the Instruction is not aggregated.

2.5.6.3.4 Raw responses

Raw responses are available for all instructions without aggregation. They may also be available for aggregated instructions if the 'KeepRaw' flag was set to true when they were issued. Because a single endpoint may return multiple response rows and there could be hundreds of thousands of connected endpoints, raw responses are designed to be retrieved using a paging mechanism. Because you don't know how may rows each endpoint has returned, you can just keep pooling Responses API.

When you ask for responses, you have to supply a start marker inside the "Start" property. If you'd like to start from the beginning (for example, the first response row) this marker should have the value of "0;0". Each time you receive responses, the object you receive will have, apart from responses themselves, a "Range" field.

The "Range" field is a marker denoting where the reading of responses finished. Next time you ask for responses, if you want to resume where you left off, supply the value you received as "Range" in "Start" property of the next call. When you get the next "page" of response you'll get another "Range" marker you can supply as "Start" in a subsequent call.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request with following payload: **Payload sent to** https://my.tachyon.server/Consumer/Responses/2016 Expand source `{ "Filter":null, "Start": "0;0", "PageSize": 20 }` Returns the following response:	Use Responses object inside the Tachyon connector instance. **Retrieving non aggregated responses in a loop** Expand source `var pageSize = 10; var startMarker = "0;0"; while (loopCondition) { var responses = connector.Responses.GetResponses(20`

Direct Consumer API call

C# code using Consumer SDK library

Return payload Expand source

```
{
    "Range": "MTsy",
    "Responses": [
        {
            "Id": 1,
            "ShardId": 1,
            "TachyonGuid": "4c4c4544-0051-5a10-80
58-b6c04f47354a",
            "Fqdn": "somemachine.somedomain",
            "Status": 0,
            "ExecTime": 113,
            "CreatedTimestampUtc": "2018-04-23T0
8:42:06.24Z",
            "ResponseTimestampUtc": "2018-04-23T0
8:42:05Z",
            "Values": {
                "Message": "hello! from somemachi
ne"
            },
            "Blob": null
        }
    ]
}
```

In the next POST call, use 'Range' from the response as value for "Start" inside the payload sent with the request:

2nd call with payload sent again to https://my.tachyon.server/Consumer/Responses/2016 Expand source

```
{
    "Filter":null,
    "Start": "MTsy",
    "PageSize": 20
}
```

to receive the following response.

Return payload Expand source

```
{
    "Range": "MTsy",
    "Responses": []
}
```

```
16, startMarker, pageSize);
   if (responses.Success)
   {
      startMarker = respons
es.ReceivedObject.Range;
      if (responses.Receive
dObject.Responses.Any())
      {
         foreach (var respo
nse in responses.ReceivedOb
ject.Responses)
         {
            // display resp
onses
         }
      }
   }
   else
   {
      foreach (var error in
responses.Errors)
      {
         // display error
         // break the loop
or wait to retry
      }
   }
   // wait a time interval
}
```

Direct Consumer API call	C# code using Consumer SDK library

In this case, there were no further responses.

2.5.6.3.5 Aggregated responses

Aggregated responses should be far fewer than raw responses (unless the aggregation has been badly designed) and they do not support pagination.

Aggregated values will be re-calculated when another agent responds. This means that examining "TotalSuccessRespondents" will give you an indication when to get the aggregated values again. If the count has changed, it means another agent has responded and the values most likely changed, so you might want to retrieve them.

You can specify pagesize in the request, but this simply limits the number of responses retrieved. There's no start marker in the request or "Range" property in the response so you cannot request the next X responses. You always have to start from the beginning because when another agent responds, the aggregated values are re-calculated so the values you have already fetched would have changed.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Responses/2018/Aggregate Expand source
 	
`{ "Filter":null, "PageSize": 20 }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Range": "0;0", "Responses": [{ "Id": 1, "ShardId": 1, "TachyonGuid": "00000000-0000-0000-0000-000000000000", "Fqdn": null, "Status": 0, "ExecTime": 0, "CreatedTimestampUtc": "0001-01-01T00:00:00Z", "ResponseTimestampUtc": "1970-01-01T00:0`
	Use Responses object inside the Tachyon connector instance.
	
	Retrieving Aggregated responses in a loop Expand source
	
	`while (loopCondition) { var responses = connector.Responses.GetAllAggregatedResponses(2018); if (responses.Success) { if (responses.ReceivedObject.Responses.Any()) { foreach (var response in responses.ReceivedObject.Responses) { // display responses } } } else {`

Direct Consumer API call

C# code using Consumer SDK library

```
0:00Z",
        "Values": {
            "Publisher": "",
            "Product": "hMailServer 5.6.6-B2383
",
            "Version": "",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 2,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000-0000-0000-
000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-01-01T00:0
0:00Z",
        "ResponseTimestampUtc": "1970-01-01T00:0
0:00Z",
        "Values": {
            "Publisher": "",
            "Product": "IcoFX 2.12",
            "Version": "2.12",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 3,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000-0000-0000-
000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-01-01T00:0
0:00Z",
        "ResponseTimestampUtc": "1970-01-01T00:0
0:00Z",
        "Values": {
            "Publisher": "",
            "Product": "IIS Express Application
Compatibility Database for x64",
            "Version": "",
```

```
      foreach (var error in
responses.Errors)
      {
         // display error
         // break the loop
or wait to retry
      }
   }
}
```

Direct Consumer API call	C# code using Consumer SDK library


```
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 4,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000-0000-0000-
000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-01-01T00:0
0:00Z",
        "ResponseTimestampUtc": "1970-01-01T00:0
0:00Z",
        "Values": {
            "Publisher": "",
            "Product": "IIS Express Application
Compatibility Database for x86",
            "Version": "",
            "Count": 1
        },
        "Blob": null
    }]
}
```


In the previous example, you can see the Fqdn field is null. This is because you're looking at aggregated data, meaning it's not coming from a single agent. Read the following sections for more details about the return object.

2.5.6.3.6 Other responses

Any response that does not yield data counts as an "Other" response. These responses are held in a separate table and do not conform to either regular or aggregation schema.

When an agent sends a reply to an instruction, the reply can have one of 5 statuses:

Numerical value	Name	Description
0	Success	Instruction executed successfully and data was returned.
1	SuccessNoContent	Instruction executed successfully but there was no data to return.

Numerical value	Name	Description
2	Error	An error occurred while executing the instruction.
3	NotImplemented	Instruction required functionality that's not implemented on the platform the Agent is running on.
4	ResponseTooLarge	Response to the instruction was too large to be sent. This is based on the Switch's settings of how large the payload can be.

Every status other than Success (0) counts as "Other" response.

You can retrieve these responses by calling:

https://my.tachyon.server/Consumer/ResponseErrors/{instructionId}/

Or in aggregated form by calling:

https://my.tachyon.server/Consumer/ResponseErrors/{instructionId}/Aggregate

Where they'll be aggregated by the Error Data. This helps finding common issues as they'll likely have the same error data.

Other responses are paginated the same way regular responses are. You use the "Range" property returned by the call as the value "Start" property of the next call, exactly the same way as you would with regular responses. Aggregated other responses are not paginated, so this behavior mimics the regular responses.

Raw other responses can be retrieved like this:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/ResponseErrors/2017 Expand source
 	
`{ "Start": "0;0", "PageSize": 20 }`
 	
Returns this response:
 	
Return payload Expand source
 	
`{ "Range": "MTs2", "Responses": [ {`
	Use ResponseErrors object inside the Tachyon connector instance.
	
	Retrieving other responses Expand source
	
	`var pageSize = 10; var startMarker = "0;0"; while (loopCondition) { var responses = connector.ResponseErrors.GetResponseErrors(2017, "0;0", pageSize) if (responses.Succ`

Direct Consumer API call

```
            "Id": 1,
            "ShardId": 1,
            "TachyonGuid": "a4f14ba5-97f5-f64c-b3ff-85
d2f5d59cee",
            "Fqdn": "somemachine1.somedomain",
            "Status": 3,
            "ExecTime": 0,
            "ErrorData": "WMI not supported",
            "CreatedTimestampUtc": "2018-04-23T13:10:4
5.9Z",
            "ResponseTimestampUtc": "2018-04-23T13:10:
45Z"
        },
        {
            "Id": 2,
            "ShardId": 1,
            "TachyonGuid": "abf0b756-1ace-3e4a-a8f5-7b
219974f77a",
            "Fqdn": "somemachine2.somedomain",
            "Status": 3,
            "ExecTime": 0,
            "ErrorData": "WMI not supported",
            "CreatedTimestampUtc": "2018-04-23T13:10:4
5.9Z",
            "ResponseTimestampUtc": "2018-04-23T13:10:
46Z"
        },
        {
            "Id": 3,
            "ShardId": 1,
            "TachyonGuid": "e28ff7ee-5d15-ec46-94f5-3a
2e5a9dde77",
            "Fqdn": "somemachine3.somedomain",
            "Status": 3,
            "ExecTime": 0,
            "ErrorData": "WMI not supported",
            "CreatedTimestampUtc": "2018-04-23T13:10:4
5.9Z",
            "ResponseTimestampUtc": "2018-04-23T13:10:
45Z"
        },
        {
            "Id": 4,
            "ShardId": 1,
            "TachyonGuid": "ae843206-f183-7242-a88c-b4
08354f2388",
            "Fqdn": "somemachine4.somedomain",
```

C# code using Consumer SDK library

```
ess)
   {
      startMarker = r
esponses.ReceivedObje
ct.Range;
      if (responses.R
eceivedObject.Respons
es.Any())
      {
         foreach (var
response in response
s.ReceivedObject.Resp
onses)
         {
            // displa
y response errors
         }
      }
   }
   else
   {
      foreach (var er
ror in responses.Erro
rs)
      {
         // display e
rror
         // break the
loop or wait to retry
      }
   }
}
```

responses.ReceivedObject object will contain the same data you can see in the Json response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
            "Status": 3,
            "ExecTime": 0,
            "ErrorData": "WMI not supported",
            "CreatedTimestampUtc": "2018-04-23T13:10:4
5.9Z",
            "ResponseTimestampUtc": "2018-04-23T13:10:
45Z"
        },
        {
            "Id": 5,
            "ShardId": 1,
            "TachyonGuid": "b1ff281e-d97d-764b-ac42-98
dee0619a93",
            "Fqdn": "somemachine5.somedomain",
            "Status": 3,
            "ExecTime": 830,
            "ErrorData": "WMI not supported",
            "CreatedTimestampUtc": "2018-04-23T13:11:0
2.123Z",
            "ResponseTimestampUtc": "2018-04-23T13:10:
53Z"
        }
    ]
}
```

And aggregates like this:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/ResponseErrors/2017/Aggregate Expand source
 	
`{ "PageSize": 20 }`
 	
will yield following response:
 	
Return payload Expand source
 	
`[ { "ErrorData": "WMI not supported", "Count": 5`
	Use ResponseErrors object inside the Tachyon connector instance.
	
	Retrieving aggregated other responses Expand source
	
	`while (loopCondition) { var responses = connector.ResponseErrors.GetAllAggregatedResponseErrors(2017); if (responses.Success) {`

Direct Consumer API call	C# code using Consumer SDK library

}
]
	if (responses.ReceivedObject.Any())
	{
	foreach (var response in responses.ReceivedObject)
	{
	// display response errors
	}
	}
	}
	else
	{
	foreach (var error in responses.Errors)
	{
	// display error
	// break the loop or wait to retry
	}
	}
	}
	
	responses.ReceivedObject object will contain the same data you can see in the JSON response on the left.

2.5.6.3.7 Examining a Response object

Raw and aggregated responses, as well as raw "other" responses, return objects similar to one another:

Property	Description
Id	Id of the response. Will be unique within the shard.
ShardId	Id of the shard this Id comes from.
TachyonGuid	Tachyon GUID of the Agent who sent the response. Not valid for aggregated responses.

Property	Description
Fqdn	Fqdn of the Agent who sent the response. Not valid for aggregated responses.
Status	Status (see table above).
ExecTime	Instruction's execution time.
CreatedTimestampUtc	Timestamp when this response was created on the Agent.
ResponseTimestampUtc	Timestamp when this response was received by the Core API.

In addition, raw and aggregated responses have two more properties:

Property	Description
Values	Collection of key value pairs, where keys are columns from the instruction's schema.
Blob	This field would hold the entire response received from Agent for instructions that do not have a schema.

raw other responses have one extra property:

Property	Description
ErrorData	Error text. This will usually be a description of the error but can be a generic text with an expanded and more readable version of the "Status" field

2.5.6.3.8 Reading responses

Previous paragraph shows the response object in its generic form, but the really interesting part of it is the "Values" property.

While “Values” will always be there what is inside it is dictated by the instruction’s schema.

The name of the schema field will be the name of the property in the response’s “Values” object, and the value of the field in the response’s Value will be the value provided by the agent.

Assuming we have an instruction with following schema:

Schema Expand source

```
[
    {
        "Name": "BIOSVersion",
        "Type": "string",
        "Length": 512
```

```
    },
    {
        "Name": "Caption",
        "Type": "string",
        "Length": 512
    },
    {
        "Name": "Manufacturer",
        "Type": "string",
        "Length": 512
    },
    {
        "Name": "PrimaryBIOS",
        "Type": "bool",
        "Length": 0
    },
    {
        "Name": "ReleaseDate",
        "Type": "datetime",
        "Length": 0
    },
    {
        "Name": "SerialNumber",
        "Type": "string",
        "Length": 512
    }
]
```

The response object will look like this:

Sample response Expand source

```
{
        "Id": 1,
        "ShardId": 1,
        "TachyonGuid": "63fc4911-f6ca-463c-9bf9-4497bf88be1f",
        "Fqdn": "CLIENT-WIN7X64.TachyonDomain.Lab",
        "Status": 0,
        "ExecTime": 15,
        "CreatedTimestampUtc": "2020-02-11T14:47:17.307Z",
        "ResponseTimestampUtc": "2020-02-11T14:47:17Z",
        "Values": {
                "BIOSVersion": "VRTUAL - 12001807, Intel(R) Xeon(R) CPU E5-2620
v3 @ 2.40GHz, BIOS Date: 12/07/18 15:46:29  Ver: 09.00.08, BIOS Date: 12/07/18
15:46:29  Ver: 09.00.08",
                "Caption": "Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz",
                "Manufacturer": "American Megatrends Inc.",
                "PrimaryBIOS": true,
                "ReleaseDate": "2018-12-07T00:00:00+00:00",
                "SerialNumber": "8546-4968-3506-2716-2087-9169-05"
        },
```

```
        "Blob": null
}
```

As you can see, there's a property inside the "Values" for each schema object and the name of that property is the same as the Name of the object in the schema.

This means that this entry in the Values object:

```
"SerialNumber": "8546-4968-3506-2716-2087-9169-05"
```

Represents this schema field:

```
{
    "Name": "SerialNumber",
    "Type": "string",
    "Length": 512
}
```

2.5.6.4 Switching from high level to detailed view of responses

Using aggregated and raw data in combination gives an opportunity to present the data as an overview with the ability to see details for the chosen aggregate if needed. This feature is commonly referred to as "drill down" in Explorer, where one can expand an aggregated row to see all the raw responses that make up the aggregated row.

Both sets of data are required for this to work so the instruction has to have "KeepRaw" set to true.

This example issues an instruction based on "1E-Explorer-TachyonCore-AllInstalledSoftware" instruction definition, which looks like:

Instruction definition 1E-Explorer-TachyonCore-AllInstalledSoftware Expand source

```
{
    "Id": 1051,
    "Name": "1E-Explorer-TachyonCore-AllInstalledSoftware",
    "Description": "Returns all installed software.",
    "InstructionSetId": 2,
    "InstructionSetName": "Everything",
    "InstructionType": 0,
    "ReadablePayload": "What software is installed?",
    "Parameters": null,
    "Schema": [
        {
            "Name": "Product",
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Publisher",
```

```
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Version",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "InstallDate",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "Architecture",
            "Type": "string",
            "Length": 20,
            "RenderAs": null
        }
    ],
    "Aggregation": {
        "Schema": [
            {
                "Name": "Publisher",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Product",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Version",
                "Type": "string",
                "Length": 128,
                "RenderAs": null
            },
            {
                "Name": "Count",
                "Type": "int64",
                "Length": 0,
                "RenderAs": null
            }
        ],
```

```
        "GroupBy": "Publisher,Product,Version",
        "Operations": [
            {
                "Name": "Count",
                "Type": "count"
            }
        ]
    },
    "InstructionTtlMinutes": 10,
    "ResponseTtlMinutes": 30,
    "MinimumInstructionTtlMinutes": 10,
    "MaximumInstructionTtlMinutes": 1440,
    "MinimumResponseTtlMinutes": 10,
    "MaximumResponseTtlMinutes": 10080,
    "Workflow": null,
    "ResponseTemplateId": 1,
    "OriginalFileName": null,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [
            {
                "Id": "mainchart",
                "Title": "Installed software - 5 most common application from 5
most common publishers",
                "Type": "Bar",
                "X": "Product",
                "Y": "Count",
                "Z": "Publisher",
                "PostProcessor": "processingFunction",
                "Size": 1,
                "Row": 1
            }
        ],
        "PostProcessors": [
            {
                "Name": "processingFunction",
                "Function": "ProcessMultiSeries('Product', 'Count', 'Publisher'
, '5', '5', 'false')"
            }
        ]
    },
    "Version": "4",
    "Author": "1E",
    "IsLicensed": true,
    "UploadedTimestampUtc": "2018-04-09T13:33:30.807Z",
    "NumberOfTimesExecuted": 4
}
```

To get the overview, request aggregated responses, then pick a row and request the raw responses used to form that row. Do this by looking at the aggregated column values and using them as a filter for the raw data request.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Responses/2020/Aggregate Expand source

```
{
   "PageSize":20
}
```

Returns this response containing aggregated responses:

Return payload with aggregated responses Expand source

```
{
    "Range": "0;0",
    "Responses": [{
        "Id": 1,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E Agent",
            "Version": "7.2.0",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 2,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
```

You can create a filter explicitly like:

Drill down example with explicit details filter Expand source

```
var pageSize = 20;
var startMarker = "0;0";

var definition = connector.Instruction
Definitions.GetInstructionDefinition(1
051);
var instruction = connector.Instructio
ns.SendInstruction(definition.Received
Object.Id, null, 60, 120);
var aggregated = connector.Responses.G
etAllAggregatedResponses(instruction.R
eceivedObject.Id);
var selectedRow = aggregated.ReceivedO
bject.Responses.FirstOrDefault(p => p.
Id == 1);
if (selectedRow != null)
{
    var responsesSearch = new Response
s
    {
        PageSize = pageSize,
        Start = startMarker,
        Filter = new ExpressionObject
        {
            Operator = "AND",
            Operands = new List<Expres
sionObject>
            {
                new ExpressionObject
                {
                    Attribute = "Publi
sher",
                    Operator = "==",
                    Value = "1E"
                },
                new ExpressionObject
                {
                    Attribute = "Produ
```

Direct Consumer API call

C# code using Consumer SDK library

```
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E NomadBranc
h x64",
            "Version": "6.3.100",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 3,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E PXE Lite L
ocal",
            "Version": "3.1.200",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 4,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
```

```
ct",
                    Operator = "==",
                    Value = "1E Agent"
                },
                new ExpressionObject
                {
                    Attribute = "Versi
on",
                    Operator = "==",
                    Value = "7.2.0"
                }
            }
        }
    };

    var detailed = connector.Response
s.GetResponses(instruction.ReceivedObj
ect.Id, responsesSearch);
}
```

or be more general by examining the group by columns from the original instruction definition schema:

Drill down example with automatic aggregated column selection for the filter Expand source

```
var pageSize = 20;
var startMarker = "0;0";

var definition = connector.Instruction
Definitions.GetInstructionDefinition(1
051);
var instruction = connector.Instructio
ns.SendInstruction(definition.Received
Object.Id, null, 60, 120);
var aggregated = connector.Responses.G
etAllAggregatedResponses(instruction.R
eceivedObject.Id);
var selectedRow = aggregated.ReceivedO
bject.Responses.FirstOrDefault(p => p.
Id == 1);
if (selectedRow != null)
{
    var filterElements = new List<Expr
essionObject>();
    var groupedColumns = definition.Re
ceivedObject.Aggregation.GroupBy.Split
```

Direct Consumer API call

```
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E Shopping A
gent",
            "Version": "1.0.200",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 5,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E Shopping C
lient Identity",
            "Version": "5.1.0",
            "Count": 1
        },
        "Blob": null
    },
    {
        "Id": 6,
        "ShardId": 1,
        "TachyonGuid": "00000000-0000
-0000-0000-000000000000",
        "Fqdn": null,
        "Status": 0,
        "ExecTime": 0,
        "CreatedTimestampUtc": "0001-
01-01T00:00:00Z",
        "ResponseTimestampUtc": "1970
-01-01T00:00:00Z",
        "Values": {
            "Publisher": "1E",
            "Product": "1E Tachyon Ag
ent x64",
            "Version": "3.1.0",
```

C# code using Consumer SDK library

```
(',');
    foreach (var column in groupedColu
mns)
    {
        filterElements.Add(new Express
ionObject
        {
            Operator = "==",
            Attribute = column,
            Value = selectedRow.Values
[column].ToString()
        });
    }

    var responsesSearch = new Response
s
    {
        PageSize = pageSize,
        Start = startMarker,
        Filter = new ExpressionObject
        {
            Operator = "AND",
            Operands = filterElements
        }
    };

    var detailed = connector.Response
s.GetResponses(instruction.ReceivedObj
ect.Id, responsesSearch);
}
```

Note

The generic approach assumes the equality operator is what you wish to use with each column.

You could also examine the definition.ReceivedObject.Aggregation.Schema collection to get all the aggregated columns. Be aware that some of these columns might contain data computed by the aggregation operation itself, like count of elements. You might not want to filter on those because they do not exist in the original schema for raw data.

Direct Consumer API call

C# code using Consumer SDK library

```
            "Count": 1
        },
        "Blob": null
    }]
}
```

From this list we will pick a row, for instance the 1E Agent (row Id 1).

From the definition we can see this instruction is aggregated on Publisher, Product and Version, so we can set up a filter for those columns and use it when requesting raw data using a POST request:

Payload sent to https://my.tachyon.server/Consumer/Responses/2020 Expand source

```
{
    "Filter": {
        "Operator": "AND",
        "Operands": [{
            "Attribute": "Publisher",
            "Operator": "==",
            "Value": "1E"
        },
        {
            "Attribute": "Product",
            "Operator": "==",
            "Value": "1E Agent"
        },
        {
            "Attribute": "Version",
            "Operator": "==",
            "Value": "7.2.0"
        }]
    },
    "Start": "0;0",
    "PageSize": 20
}
```

which returns raw responses matching the filter. These responses will be the rows used to form the aggregated row we picked.

Return payload with filtered raw responses Expand source

To mitigate you could correlate the aggregation schema:

- With the regular schema to make sure you're only searching for columns that do exist in the regular schema
- Columns with the aggregation operations list - each operation produces a column and it's these columns you do not want to filter on.

Direct Consumer API call | C# code using Consumer SDK library


```
{
    "Range": "MTsyNQ==",
    "Responses": [{
        "Id": 24,
        "ShardId": 1,
        "TachyonGuid": "4c4c4544-0051
-5a10-8058-b6c04f47354a",
        "Fqdn": "SomeDomain\\SomeMach
ine",
        "Status": 0,
        "ExecTime": 2991,
        "CreatedTimestampUtc": "2018-
04-24T07:53:49.81Z",
        "ResponseTimestampUtc": "2018
-04-24T07:53:48Z",
        "Values": {
            "Product": "1E Agent",
            "Publisher": "1E",
            "Version": "7.2.0",
            "InstallDate": "2017-09-0
6T00:00:00+00:00",
            "Architecture": "x64"
        },
        "Blob": null
    }]
}
```


2.5.6.4.1 Processed responses

The last way of retrieving responses is using processed responses.

By making a GET request to https://my.tachyon.server/Consumer/Responses/Processed/{InstructionId} endpoint you can retrieve responses that have been processed according to the rules defined in Instruction definition's ResponseTemplateConfiguration settings.

Explorer uses this feature and aims at providing data in a readily consumable way by the visualization engine the Explorer uses. Read more about visualizations in custom response visualizations.

2.5.6.5 Cancelling and re-running instructions

Instructions can be cancelled if they have been issued but have not started (for example, because they have not been approved or authenticated) or when they are live (when they're gathering responses).

If the instruction is cancelled before it goes live, it will not be sent to agents and there will be no results to view. There'll still be an entry for this instruction so you'll be able to see what it was.

If the instruction has gone live you, you can stop the instruction instead of cancelling it and you'll have the option to either discard or keep the gathered results. If you choose to:

- Discard the results - the instruction goes to the "Cancelled" state, which is similar to "Expired" because responses are no longer available
- Keep the responses - the instruction goes to the "Cancelling" state, which is similar to "Complete" in a way that responses are still available for the duration of ResponseTtl of the instruction.

To cancel or stop an instruction, make a POST request to https://my.tachyon.server/Consumer/Instructions/{id}/cancel/{keepData}, where id is the Id of the instruction that's to be cancelled and keepData is a boolean flag indicating whether you want to keep or discard responses. The default is to "false", which means discard the responses.

Direct Consumer API call	C# code using Consumer SDK library

POST request to https://my.tachyon.server/Consumer/Instructions/2020/cancel/true without any payload.
 	
There's no response payload and success HTTP status code will mean the instruction was cancelled.
	Use Instructions object inside the Tachyon connector instance.
	
	Cancelling an instruciton
	Expand source
	
	`var result = connector.Instructions.CancelInstruction(2020, true);`
	`if (result.Success)`
	`{`
	`    // instruction was cancelled`
	`}`

Both cancelling and stopping use the same endpoint. The difference is purely to the state the instruction is in when the cancellation occurs, Explorer will show this action as either Cancel or Stop, which is why we're using these terms in this document.

2.5.6.5.1 Re-running instructions

You can rerun instructions that are not live and not pending authentication or authorization. This causes the system to issue a new instruction that is a copy of the instruction you're re-running. This will include things like scope, parameters and filters.

> **Note**
>
> Only questions can be rerun.

To rerun an instruction make a POST request to https://my.tachyon.server/Consumer/Instructions/{id}/rerun where id is the Id of the instruction you want to run again.

Direct Consumer API call	C# code using Consumer SDK library
POST request to https://my.tachyon.server/Consumer/Instructions/2020/rerun without any payload to return this response: **Payload sent to** https://my.tachyon.server/Consumer/Instructions/2020/rerun Expand source	Use Instructions object inside the Tachyon connector instance. **Re-running an instruction** Expand source

```
{
    "Id": 2022,
    "Sequence": 0,
    "Name": "1E-Explorer-TachyonCore-AllInstalledSoftware",
    "Description": "Returns all installed software.",
    "InstructionType": 0,
    "ReadablePayload": "What software is installed?",
    "Cmd": "SendAll",
    "Schema": [
        {
            "Name": "Product",
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Publisher",
            "Type": "string",
            "Length": 512,
            "RenderAs": null
        },
        {
            "Name": "Version",
            "Type": "string",
            "Length": 128,
            "RenderAs": null
        },
        {
            "Name": "InstallDate",
            "Type": "datetime",
            "Length": 0,
            "RenderAs": null
        },
        {
            "Name": "Architecture",
            "Type": "string",
```

```
var rerunInstruction = connector.Instructions.RerunInstruction(2020);
if (rerunInstruction.Success)
{
    // rerunInstruction.ReceivedObject has the newly issued instruction
}
else
{
    foreach (var error in rerunInstruction.Errors)
    {
        // display error
        // break the loop or wait to retry
    }
}
```

rerunInstruction.ReceivedObject object will contain the same data you can see in the JSON response on the left.

```
            "Length": 20,
            "RenderAs": null
        }
    ],
    "Aggregation": {
        "Schema": [
            {
                "Name": "Publisher",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Product",
                "Type": "string",
                "Length": 512,
                "RenderAs": null
            },
            {
                "Name": "Version",
                "Type": "string",
                "Length": 128,
                "RenderAs": null
            },
            {
                "Name": "Count",
                "Type": "int64",
                "Length": 0,
                "RenderAs": null
            }
        ],
        "GroupBy": "Publisher,Product,Version",
        "Operations": [
            {
                "Name": "Count",
                "Type": "count"
            }
        ]
    },
    "KeepRaw": true,
    "Scope": null,
    "InstructionTtlMinutes": 10,
    "ResponseTtlMinutes": 30,
    "CreatedTimestampUtc": "2018-04-25T13:22:49.983Z
",
    "SentTimestampUtc": "2018-04-24T07:53:45.217Z",
    "Status": 0,
```

```
    "WorkflowState": 0,
    "StatusTimestampUtc": "2018-04-24T08:33:46.43Z",
    "Export": false,
    "ExportLocation": "",
    "ParentInstructionId": null,
    "InstructionDefinitionId": 1051,
    "CreatedBy": "1E\\Michael.Grzywnowicz",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "ResponseTemplateId": 1,
    "ResponseTemplateConfiguration": {
        "Name": "default",
        "TemplateConfigurations": [
            {
                "Id": "mainchart",
                "Title": "Installed software - 5 mos
t common application from 5 most common publishers",
                "Type": "Bar",
                "X": "Product",
                "Y": "Count",
                "Z": "Publisher",
                "PostProcessor": "processingFunction
",
                "Size": 1,
                "Row": 1
            }
        ],
        "PostProcessors": [
            {
                "Name": "processingFunction",
                "Function": "ProcessMultiSeries('Pro
duct', 'Count', 'Publisher', '5', '5', 'false')"
            }
        ]
    },
    "Workflow": "{\"StateMachine\":\"State\"}",
    "Comments": null,
    "ScheduledInstructionId": null,
    "ApprovalOffloaded": false,
    "ActionedBy": null,
```

```
    "ActionReason": null
}
```

Return payload contains details of newly issued instruction.

Remember that the new instruction will be an exact copy of the instruction that was re-run. If you want to change anything, you'll have to issue a new instruction as seen above.

2.5.6.6 Pseudo-code examples

This section brings together the previous examples with pseudo-code examples of how to do certain things.

These examples describe a concept of how things can be done, and are not designed to be copy-pasted into working code. You should be able to copy and adjust them to the language you're using.

2.5.6.6.1 Example 1: Find a machine with specific hardware and perform an action on it.

Company XYZ, who makes network adapters, has announced that a driver for one of their cards called "GreatNIC" has an issue. You want to find all computers in your estate using that model of network adapter and update those computers by installing an updated driver.

You use the following instructions:

- The question which finds network adapters has the Id of 1 and returns a number of columns, among which are Manufacturer and Model
- The action which installs the driver has the Id of 2 and takes one parameter with the pattern of %DRIVERPATH%.

This example assumes that the instructions will run their course and will be authenticated and will not be cancelled or rejected in approval.

Expand source

```
{
    var instruction = new instance of Instruction payload
    {
        Id = 1,
        InstructionTtlMinutes = 10,
        ResponseTtlMinutes = 60,
        ResultsFilter = {
                Operator = "AND",
                Operands = [{
                        Operator = "==",
                        Attribute = "Manufacturer",
                        Value = "XYZ"
                    },
                    {
```

```
                    Operator = "==",
                    Attribute = "Model",
                    Value = "GreatNIC"
                }
            ]
        }
    }

    var sentInstruction = send the 'instruction' object to Tachyon's Instructio
ns API via a POST and receive response payload with information about issued in
struction.
    sleep for 10 minutes
    // at this point, the instruction should have finished.
    var statistics = get instruction's statistics from Tachyon InstructionStati
stics API via a GET and use sentInstruction.Id as instruciton id.
    // we will assume that no successful response means no devices have the net
work card we're looking for.
    if (statistics.TotalSuccessRespondents == 0)
        return

    var responseSearch = new instance of object used to retrieve responses
    {
        Start = "0;0",
        PageSize = statistics.TotalSuccessRespondents //we'll assume this numbe
r is fairly low and use it to get every result on one page. This assumes that e
ach Client sends back only 1 row of data.
    }
    var responses = get responses from Tachyon's Responses API via a POST and u
se sentInstruction.Id as instruction Id.
    // present a list of devices that will have their drivers updated to a user
    var followUpInstruction = new instance of Instruction payload
    {
        Id = 2,
        ParentInstructionId = sentInstruction.Id,
        InstructionTtlMinutes = 1440,
        ResponseTtlMinutes = 60 minutes,
        Parameters = [{
            Name = "driver",
            Pattern = "%DRIVERPATH%",
            Value = "\\someshare\somefolder\newdriver.exe"
        }]
    }
    var sendAction = send the 'followUpInstruction' object to Tachyon's Instruc
tions API via a POST and receive response payload with information about issued
instruction.
    // because this is an action, it will most likely require authentication an
d approval, so we should wait until those have been done and instruction is liv
e.
        while instruction state is not 5
    {
```

```
        var followupAction = get the instruction with the id of sendAction.Id
                if followupAction.Status is 5 then break loop
                else sleep for 1 minute // this is just an arbitrary time inter
val, which should be adjusted as needed.
    }
    while instruction is live //in other words, loop for the duration of Instru
ctionTtlMinutes
    {
        var start = "0;0"
        var responseSearch = new instance of object used to retrieve responses
        {
            Start = start,
            PageSize = 25 //we'll use a hard-coded page of 25, but this can be
adjusted if needed
        }
        var actionResponses = get responses from Tachyon's Responses API via a
POST and use sentAction.Id as instruction Id.
        // present the list of devices that had their drivers updated to the us
er
        start = actionResponses.Range
        sleep for 2 minutes // this is just an arbitrary time interval, which s
hould be adjusted as needed.
        check loop condition by examining if time now is greater than instructi
on's send time plus instruction ttl.
    }
}
```

2.5.6.6.2 Example 2: Retrieving aggregated responses

Assuming that we have issued an instruction with the Id of 12, here's how one could program retrieval of aggregated responses.

Expand source

```
bool isInstructionLive = false
while isInstructionLive is false
{
    var instruction = get instruction with the Id of 12
    if instruction status is 5
    {
        isInstructionLive = true
    }
    else
    {
        sleep for 60 seconds // this is just an arbitrary time interval, which
should be adjusted as needed.
    }
}

bool shouldContinue = true
var receivedCount = 0
```

```
while shouldContinue is true
{
    var instructionNow = get instruction with the Id of 12
    stats = get statistics for the instruction with the Id of 12
    if stats.TotalSuccessRespondents is greater then receivedCount
    {
        // Last count of received responses was lower then new count. Fetching
aggregated responses.
        var responses = get aggregated responses for the instruction with the I
d of 12
        receivedCount = stats.TotalSuccessRespondents
        // process responses
    }

    if instructionNow.Status is not 5 //if the instruction status has changed f
rom 5 to something else because the instruction ttl elapsed or the instruction
has been cancelled.
    {
        var responses = get aggregated responses for the instruction with the I
d of 12
        shouldContinue = false
        // process responses
        break while loop
    }

    if shouldContinue is true
        sleep for 60 seconds // this is just an arbitrary time interval, which
should be adjusted as needed.
}
```

The basis for this algorithm is that when one is retrieving aggregated responses, the aggregated values can only change when a new device/client sends a response, because this causes the aggregated values to change.

We're not using pagination here because aggregated responses do not support pagination.

2.5.6.6.3 Example 3: Retrieving raw responses

Assuming that we have issued an instruction with the Id of 12, and it either has no aggregation of has aggregation but the KeepRaw flag was set to true when the instruction was issued, here's how one could program retrieval of raw responses.

Expand source

```
bool isInstructionLive = false
while isInstructionLive is false
{
    var instruction = get instruction with the Id of 12
    if instruction status is 5
    {
        isInstructionLive = true
```

```
    }
    else
    {
        sleep for 60 seconds // this is just an arbitrary time interval, which
should be adjusted as needed.
    }
}

bool shouldContinue = true
var marker = "0;0" // This is a marker telling the Consumer API to start from t
he beginning
while shouldContinue is true
{
    var responseSearch = new instance of object used to retrieve responses
    {
        Start = marker,
        PageSize = 25 // this is just an arbitrary value, which should be adjus
ted as needed
    }
    var responses = get raw responses from Tachyon's Responses API via a POST,
send responseSearch object as payload and use 12 as instruction Id.
    marker = response.Range
    // process responses

    var instructionNow = get instruction with the Id of 12
    if instructionNow.Status is not 5 //if the instruction status has changed f
rom 5 to something else because the instruction ttl elapsed or the instruction
has been cancelled.
    {
        shouldContinue = false
        // instruction has finished
        break while loop
    }

    if shouldContinue is true
        sleep for 60 seconds // this is just an arbitrary time interval, which
should be adjusted as needed.
}
```

The above example is a scaffoling for retrieving raw responses. There is a crucial point in this example - at the point with the comment saying "instruction has finished".

At this point you have gone through the response retrieval at least once and the instruction is no longer live. This means that there might be responses you still haven't retrieved.

You could just get all responses in one go by changing the pageSize in responseSearch to sufficiently large number (for instance in C# you could use int.MaxValue) and making one last call to retrieve any remaining responses. This, however, may return a large dataset so it is usually preferred to use pagination.

Expand source

```
responseSearch = new instance of object used to retrieve responses
{
    Start = marker,
    PageSize = maximum value supported by integer
}
var responses = get raw responses from Tachyon's Responses API via a POST, send
responseSearch object as payload and use 12 as instruction Id.
// process responses
```

You also could keep the pagesize as-is and continue looping getting more and more responses up until the point where you receive fewer responses than the page can hold:

Expand source

```
var fullPage = true
while fullPage is true
{
    var responses = get raw responses from Tachyon's Responses API via a POST,
send responseSearch object as payload and use 12 as instruction Id.
    marker = response.Range
    if number of entries responses collection is lower than the page size reque
sted // in this example hard-coded to 25
        fullPage = false
    // process responses

    //You could add a sleep here, but since there are no new responses coming i
n from agents it might be preferred to simply retrieve and process responsese a
page at a time without delay.
}
```

in both cases, the code would be called where the "instruction has finished" comment is located.

Another option is to exit the code completely and loop through all responses again, startign from the beginning, or even retrieve them all in one go.

Which approach to choose depends on your particular use-case.

2.5.7 Set up in instruction to be executed on a schedule

This page describes how Tachyon instructions can be set up to execute on the schedule.

You should also read Issue an instruction, track its progress and retrieve responses page, which covers tracking instruction's progress, retrieving responses and the like, because here we will only focus on the scheduling aspect.

The C# examples assume you're using Tachyon Consumer SDK and you already have a correctly instantiated instance of Tachyon connector class in an object called connector.

All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API. In the following examples this detail is left out, stating that the returned object contains the data. For example, when we say that XYZ object contains certain data, this means the ReceivedObject contains that data, since that's always true.

2.5.7.1 What is a "scheduled instruction"

A "scheduled instruction" is a regular Tachyon instruction definition that has been configured to execute periodicaly on a specific schedule."Scope" property to the schedule creating payload

When a schedule is set up, you define the instruction as you would normally (see Issue an instruction, track its progress and retrieve responses), with parameters, scope etc. and add information about the schedule.

When the time comes to execute the schedule, Tachyon will issue a new copy of the instruction. This will be a carbon copy of the instruction that was scheduled - all fields will be copied, including auxiliary ones like Comments and ConsumerCustomData. This means that each copy of the instruction issued on a schedule will have the same parameters, the same scope or target devices etc.

2.5.7.2 Creating a schedule

To create a schedule you have to provide the template for the instruction that will be issued when the schedule is triggered and information about the schedule itself.

It is worth noting that creating a schedule will not result in an instruction being issued immediatelly. This is because instructions are issued only when a schedule is triggered.

A schedule should be created with care as it is possible to create one that will flood the system with requests. It is not advised to create a schedule where a new instruction is issued before the previous' ones instruction ttl expires.

Because Tachyon's scheduling mechanism is relatively simple, if you require features like variation in instruction parameters or targeting between instructions triggered on a schedule, you might want to look at implementing scheduling yourself and issuing regular instructions instead.

2.5.7.2.1 Core of the scheduled instruction

The core of the scheduled instruction is much the same as a regular instruction, as seen in Issue an instruction, track its progress and retrieve responses.

Each instruction issued on the schedule will be a copy of the instruction provided when a schedule is created.

2.5.7.2.2 Scheduling properties

Schedule properties with the exception of ScheduleEnabled are based on Microsoft SQL Server system schedules as described in details here: https://docs.microsoft.com/en-us/sql/relational-databases/system-tables/dbo-sysschedules-transact-sql?view=sql-server-ver15

Schedule property name	Microsoft SQL column name	Description
ScheduleEnabled	enabled	This field defined if a schedule is enabled or not. Schedules that are not enabled will not be executed. If a schedule is enabled after some of the occurrences should have happened, the schedule will resume from the

Schedule property name	Microsoft SQL column name	Description
		next scheduled occurence and will not "catch up".
ScheduleFreqType	freq_type	How frequently a job runs for this schedule. • Value of 1 means only there will be only one activation of this schedule so only one instruction will be issued • Value of 4 means daily activation • Value of 8 means weekly activation • Value of 16 means monthly activation • Value of 32 means monthly activation relative to ScheduleFreqInterval
ScheduleFreqInterval	freq_interval	Meaning of the value of this fields depends on the value of ScheduleFreqType. • If ScheduleFreqType is set to 1 (one activation of the schedule), this field is not used and should have the value of 0 • If ScheduleFreqType is set to 4, this field will have the number of days between activations of the schedule. For example, to make a schedule activate every day, set this to 1. To do it every other day, set it to 2. • If ScheduleFreqType is set to 8, this field becomes a bit flag

Schedule property name	Microsoft SQL column name	Description

that contains information on which days of the week the schedule should be activated. The values are as follows:

Day	Value
Sunday	1
Monday	2
Tuesday	4
Wednesday	8
Thursday	16
Friday	32
Saturday	64

For example, if you wanted to set the schedule up to activate every day of the week the value of ScheduleFreqInterval would be 127. If you only wanted the schedule to activate on Mondays and Fridays it would be 34.

- If ScheduleFreqType is set to 16, the value of this field will be the number of the day of the month on which the schedule will be activated.
- If ScheduleFreqType is set to 32, this propert will have one of following values:

<table>
<tr><th>Schedule property name</th><th>Microsoft SQL column name</th><th>Description</th></tr>
<tr><td></td><td></td><td><table>
<tr><th>Description</th><th>Value</th></tr>
<tr><td>Sunday</td><td>1</td></tr>
<tr><td>Monday</td><td>2</td></tr>
<tr><td>Tuesday</td><td>3</td></tr>
<tr><td>Wednesday</td><td>4</td></tr>
<tr><td>Thursday</td><td>5</td></tr>
<tr><td>Friday</td><td>6</td></tr>
<tr><td>Saturday</td><td>7</td></tr>
<tr><td>Day</td><td>8</td></tr>
<tr><td>Weekday</td><td>9</td></tr>
<tr><td>Weekend day</td><td>10</td></tr>
</table></td></tr>
<tr><td>ScheduleFreqSubdayType</td><td>freq_subday_type</td><td>Units for the ScheduleFreqSubdayInterval.
Value of 1 means activate at the specified time
Value of 2 means the unit is Seconds
Value of 4 means the unit is Minutes
Value of 8 means the unit is Hours</td></tr>
<tr><td>ScheduleFreqSubdayInterval</td><td>freq_subday_interval</td><td>Number of ScheduleFreqSubdayType to occur between each activation of the schedule.</td></tr>
</table>

<table>
<tr><th>Schedule property name</th><th>Microsoft SQL column name</th><th>Description</th></tr>
<tr><td>ScheduleFreqRelativeInterval</td><td>freq_relative_interval</td><td>If ScheduleFreqInterval is set to 32 this property can have one of following values:<table><tr><th>Description</th><th>Value</th></tr><tr><td>ScheduleFreqRelativeInterval is not used</td><td>0</td></tr><tr><td>First</td><td>1</td></tr><tr><td>Second</td><td>2</td></tr><tr><td>Third</td><td>4</td></tr><tr><td>Fourth</td><td>8</td></tr><tr><td>Last</td><td>16</td></tr></table>If ScheduleFreqInterval is set to any other value this field is ignored.</td></tr>
<tr><td>ScheduleFreqRecurrenceFactor</td><td>freq_recurrence_factor</td><td>Number of weeks or months between the activation of the schedule. ScheduleFreqRecurrenceFactor is used only if ScheduleFreqType is set to 8, 16, or 32. If this column contains 0, ScheduleFreqRecurrenceFactor will not be used.</td></tr>
<tr><td>ScheduleActiveStartDate</td><td>active_start_date</td><td>Date on which the schedule should begin formatted as YYYYMMDD</td></tr>
<tr><td>ScheduleActiveEndDate</td><td>active_end_date</td><td>Date on which the schedule should finish formatted as YYYYMMDD</td></tr>
<tr><td>ScheduleActiveStartTime</td><td>active_start_time</td><td>Time between ScheduleActiveStartDate and ScheduleActiveEndDate which will be</td></tr>
</table>

Schedule property name	Microsoft SQL column name	Description
		used as starting time for the schedule exexution on a given day. Time is formatted HHMMSS, using a 24-hour clock.
ScheduleActiveEndTime	active_end_time	Time between ScheduleActiveStartDate and ScheduleActiveEndDate which will be used as stopping time for the schedule exexution on a given day. Time is formatted HHMMSS, using a 24-hour clock.

A schedule can be created disabled by setting the ScheduleEnabled property to false. This will mean that it will not be triggered and no instructions will be issued until the schedule is enabled. If the schedule is enabled at a point in time where it would have already missed some activations, it will simply resume from the next scheduled activation. For example, if on Monday morning you create a disabled schedule that is meant to trigger every day at mid day and then go ahead an enabled it on Wednesday at 5pm, the next triggered activation will be on Thursday at mid day, and the schedule would have "missed" activations on Monday, Tuesday and Wednesday because it was disabled. This means that an instruction will be issued at mid day Thursday, but there will be no instructions for Monday, Tuesday or Wednesday.

2.5.7.3 Managing schedules

In this section we will learn how to examine what schedules are available, how to create, modify and cancel them and how to see what instructions will be issued by a schedule.

2.5.7.3.1 Retrieving scheduled instructions

To retrieve all scheduled instructions (but not instructions issued by schedules) you should use the https://my.tachyon.server/Consumer/ScheduledInstructions endpoint.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/ScheduledInstructions will return all scheduled instructions:	Use ScheduledInstructions object inside the Tachyon connector instance.
Return payload Expand source	**Retrieving all scheduled instructions** Expand source

```
[
    {
        "Id": 1,
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"Name": "1E-Explorer-TachyonAgent-Echo ",
"Cmd": "SendAll",
"KeepRaw": true,
"Scope": null,
"ReadableScope": null,
"InstructionTtlMinutes": 60,
"ResponseTtlMinutes": 120,
"CreatedTimestampUtc": "2019-12-06T10:53:31.903Z",
"Export": false,
"ExportLocation": null,
"ParentInstructionId": null,
"InstructionDefinitionId": 23,
"CreatedBy": "SomeDomain\\Jane.Doe",
"ResultsFilter": null,
"PreviousResultsFilter": null,
"Devices": null,
"ConsumerId": 1,
"ConsumerName": "Explorer",
"ConsumerCustomData": null,
"ParameterJson": "[{\"Name\":\"msg\",\"Pattern\":\"%msg%\",\"DataType\":null,\"ControlType\":null,\"ControlMetadata\":null,\"Placeholder\":null,\"DefaultValue\":null,\"Validation\":null,\"Value\":\"Machine reporting\",\"HintText\":null,\"Source\":null}]",
"OffloadResponses": false,
"RequestedFor": null,
"Comments": "Reporting machine echo",
"ActionedBy": "Ollanius Pius",
"ActionReason": null,
"Status": 7,
"StatusTimestampUtc": "2019-12-10T13:24:35.917Z",
"Workflow": "{\"StateMachine\":\"Scheduled Instruction State with Authentication and Approval\",\"Authenticating\":\"Scheduled Instruction Authentication\",\"Approval\":\"Scheduled Instruction Approval\",\"InProgress\":\"Scheduled Instruction InProgress\",\"Cancelling\":\"Scheduled Instruction Cancelling\"}",
"WorkflowState": 8,
"ApprovalOffloaded": false,
"ReadablePayload": "Echo message %msg%. ",
"ScheduleEnabled": true,</pre>
	<pre>schedules = connector.ScheduledInstructions.GetAll();</pre>
	"schedules" object will contain the same data you can see in the JSON response on the left.

```
        "ScheduleFreqType": 4,
        "ScheduleFreqInterval": 1,
        "ScheduleFreqSubdayType": 8,
        "ScheduleFreqSubdayInterval": 12,
        "ScheduleFreqRelativeInterval": 0,
        "ScheduleFreqRecurrenceFactor": 0,
        "ScheduleActiveStartDate": "2019-12-06T
00:00:00Z",
        "ScheduleActiveEndDate": "2019-12-09T0
0:00:00Z",
        "ScheduleActiveStartTime": 120000,
        "ScheduleActiveEndTime": 120000,
        "ScheduleLastExecuted": null,
        "ScheduleNextExecution": null,
        "ScheduleReadableFrequency": "Occurs ev
ery 1 day(s) every 12 hours starting at 12:00:0
0 ending at 12:00:00 starting on 06/12/2019 end
ing on 09/12/2019"
    },
    {
        "Id": 2,
        "Name": "1E-Explorer-TachyonAgent-Echo
",
        "Cmd": "SendAll",
        "KeepRaw": true,
        "Scope": null,
        "ReadableScope": null,
        "InstructionTtlMinutes": 60,
        "ResponseTtlMinutes": 120,
        "CreatedTimestampUtc": "2019-12-10T13:0
4:06.6Z",
        "Export": false,
        "ExportLocation": null,
        "ParentInstructionId": null,
        "InstructionDefinitionId": 23,
        "CreatedBy": "SomeDomain\\Jane.Doe",
        "ResultsFilter": null,
        "PreviousResultsFilter": null,
        "Devices": null,
        "ConsumerId": 1,
        "ConsumerName": "Explorer",
        "ConsumerCustomData": null,
        "ParameterJson": "[{\"Name\":\"msg\",\"
Pattern\":\"%msg%\",\"DataType\":\"string\",\"C
ontrolType\":\"freeText\",\"ControlMetadata\":n
ull,\"Placeholder\":\"message to echo\",\"Defau
ltValue\":null,\"Validation\":{\"Regex\":null,\
```

```
"MaxLength\":\"200\",\"AllowedValues\":null,\"N
umValueRestrictions\":null},\"Value\":\"echo ec
ho echo\",\"HintText\":null,\"Source\":null}]",
        "OffloadResponses": false,
        "RequestedFor": null,
        "Comments": null,
        "ActionedBy": "Ollanius Pius",
        "ActionReason": null,
        "Status": 7,
        "StatusTimestampUtc": "2019-12-12T13:2
7:57.5Z",
        "Workflow": "{\"StateMachine\":\"Schedu
led Instruction State with Authentication and A
pproval\",\"Authenticating\":\"Scheduled Instru
ction Authentication\",\"Approval\":\"Scheduled
Instruction Approval\",\"InProgress\":\"Schedul
ed Instruction InProgress\",\"Cancelling\":\"Sc
heduled Instruction Cancelling\"}",
        "WorkflowState": 8,
        "ApprovalOffloaded": false,
        "ReadablePayload": "Echo message %msg%.
",
        "ScheduleEnabled": true,
        "ScheduleFreqType": 1,
        "ScheduleFreqInterval": 0,
        "ScheduleFreqSubdayType": 0,
        "ScheduleFreqSubdayInterval": 0,
        "ScheduleFreqRelativeInterval": 0,
        "ScheduleFreqRecurrenceFactor": 0,
        "ScheduleActiveStartDate": "2019-12-10T
00:00:00Z",
        "ScheduleActiveEndDate": "2019-12-11T0
0:00:00Z",
        "ScheduleActiveStartTime": 142000,
        "ScheduleActiveEndTime": 140200,
        "ScheduleLastExecuted": "2019-12-10T14:
20:00Z",
        "ScheduleNextExecution": null,
        "ScheduleReadableFrequency": "Occurs on
ce on 10/12/2019 at 14:20:00"
    },
    {
        "Id": 3,
        "Name": "1E-Explorer-TachyonAgent-Echo
",
        "Cmd": "SendAll",
        "KeepRaw": true,
```

```
        "Scope": null,
        "ReadableScope": null,
        "InstructionTtlMinutes": 60,
        "ResponseTtlMinutes": 120,
        "CreatedTimestampUtc": "2019-12-10T13:0
6:47.78Z",
        "Export": false,
        "ExportLocation": null,
        "ParentInstructionId": null,
        "InstructionDefinitionId": 23,
        "CreatedBy": "SomeDomain\\Jane.Doe",
        "ResultsFilter": null,
        "PreviousResultsFilter": null,
        "Devices": null,
        "ConsumerId": 1,
        "ConsumerName": "Explorer",
        "ConsumerCustomData": null,
        "ParameterJson": "[{\"Name\":\"msg\",\"
Pattern\":\"%msg%\",\"DataType\":\"string\",\"C
ontrolType\":\"freeText\",\"ControlMetadata\":n
ull,\"Placeholder\":\"message to echo\",\"Defau
ltValue\":null,\"Validation\":{\"Regex\":null,\
"MaxLength\":\"200\",\"AllowedValues\":null,\"N
umValueRestrictions\":null},\"Value\":\"echo ec
ho echo\",\"HintText\":null,\"Source\":null}]",
        "OffloadResponses": false,
        "RequestedFor": null,
        "Comments": null,
        "ActionedBy": "Ollanius Pius",
        "ActionReason": null,
        "Status": 7,
        "StatusTimestampUtc": "2019-12-12T13:2
7:59.543Z",
        "Workflow": "{\"StateMachine\":\"Schedu
led Instruction State with Authentication and A
pproval\",\"Authenticating\":\"Scheduled Instru
ction Authentication\",\"Approval\":\"Scheduled
Instruction Approval\",\"InProgress\":\"Schedul
ed Instruction InProgress\",\"Cancelling\":\"Sc
heduled Instruction Cancelling\"}",
        "WorkflowState": 8,
        "ApprovalOffloaded": false,
        "ReadablePayload": "Echo message %msg%.
",
        "ScheduleEnabled": true,
        "ScheduleFreqType": 1,
        "ScheduleFreqInterval": 0,
```

Direct Consumer API call	C# code using Consumer SDK library

```
        "ScheduleFreqSubdayType": 0,
        "ScheduleFreqSubdayInterval": 0,
        "ScheduleFreqRelativeInterval": 0,
        "ScheduleFreqRecurrenceFactor": 0,
        "ScheduleActiveStartDate": "2019-12-10T
00:00:00Z",
        "ScheduleActiveEndDate": "2019-12-11T0
0:00:00Z",
        "ScheduleActiveStartTime": 142000,
        "ScheduleActiveEndTime": 140200,
        "ScheduleLastExecuted": "2019-12-10T14:
20:00Z",
        "ScheduleNextExecution": null,
        "ScheduleReadableFrequency": "Occurs on
ce on 10/12/2019 at 14:20:00"
    }
]
```

You can also retrieve a single scheduled instruction by using its Id. You cannot use the name because there might be multiple scheduled instructions that use the same instruction definition.

It is important to remember that this is the Id of the schedule, not the instruction definition on which scheduled instructions are based upon. This example shows how you can locate all schedules based on a specific instruction definition.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/ScheduledInstructions/1 will yield following response:	Use ScheduledInstructions object inside the Tachyon connector instance.
Return payload Expand source	**Retrieving a single scheduled instruction** Expand source
`{ "Id": 1, "Name": "1E-Explorer-TachyonAgent-Echo", "Cmd": "SendAll", "KeepRaw": true, "Scope": null, "ReadableScope": null, "InstructionTtlMinutes": 60, "ResponseTtlMinutes": 120, "CreatedTimestampUtc": "2019-12-06T10:53:31.903Z", "Export": false, "ExportLocation": null, "ParentInstructionId": null,`	`schedule = connector.ScheduledInstructions.GetScheduledInstruction(1);` "schedule" object will contain the same data you can see in the JSON response on the left.

```
    "InstructionDefinitionId": 23,
    "CreatedBy": "SomeDomain\\Jane.Doe",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "Devices": null,
    "ConsumerId": 1,
    "ConsumerName": "Explorer",
    "ConsumerCustomData": null,
    "ParameterJson": "[{\"Name\":\"msg\",\"Pa
ttern\":\"%msg%\",\"DataType\":null,\"Control
Type\":null,\"ControlMetadata\":null,\"Placeh
older\":null,\"DefaultValue\":null,\"Validati
on\":null,\"Value\":\"Machine reporting\",\"H
intText\":null,\"Source\":null}]",
    "OffloadResponses": false,
    "RequestedFor": null,
    "Comments": "Reporting machine echo",
    "ActionedBy": "Ollanius Pius",
    "ActionReason": null,
    "Status": 7,
    "StatusTimestampUtc": "2019-12-10T13:24:3
5.917Z",
    "Workflow": "{\"StateMachine\":\"Schedule
d Instruction State with Authentication and A
pproval\",\"Authenticating\":\"Scheduled Inst
ruction Authentication\",\"Approval\":\"Sched
uled Instruction Approval\",\"InProgress\":\"
Scheduled Instruction InProgress\",\"Cancelli
ng\":\"Scheduled Instruction Cancelling\"}",
    "WorkflowState": 8,
    "ApprovalOffloaded": false,
    "ReadablePayload": "Echo message %msg%.",
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 8,
    "ScheduleFreqSubdayInterval": 12,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 0,
    "ScheduleActiveStartDate": "2019-12-06T0
0:00:00Z",
    "ScheduleActiveEndDate": "2019-12-09T00:0
0:00Z",
    "ScheduleActiveStartTime": 120000,
    "ScheduleActiveEndTime": 120000,
    "ScheduleLastExecuted": null,
    "ScheduleNextExecution": null,
    "ScheduleReadableFrequency": "Occurs ever
```

Direct Consumer API call	C# code using Consumer SDK library

```
y 1 day(s) every 12 hours starting at 12:00:0
0 ending at 12:00:00 starting on 06/12/2019 e
nding on 09/12/2019"
}
```

2.5.7.3.2 Retrieving past and/or planned instructions

You can search for schedule activations that match certain parameters that have occurred or will occur between certain points in time.

Before we dwell into examples of how this can be used, let's look at how the API call itelf will look like so that in subsequent examples we can focus on crafting just the search parameters payload to handle specific use-case.

For now, let's locate all scheduled activations for a Consumer whose Id is 7 that will occur in the next week:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/ScheduledInstructions/Search
 	
Request payload Expand source
 	
`{ "Filter": { "Attribute": "ConsumerId", "Operator": "==", "Value": "7" }, "StartDate": "2018-02-14T15:10:22.935Z", "EndDate": "2020-02-21T15:10:22.936Z", "PageSize": 100 }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "TotalCount": 2, "Items": [ { "Id": 9, "Name": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails", "CreatedTimestampUtc": "2020-02-01T09:45:11.377Z",`
	Use ScheduledInstructions object inside the Tachyon connector instance.
	
	Scheduled activations in the next 7 days Expand source
	
	`var payload = new ScheduledSearch { PageSize = 100, Filter = new ExpressionObject { Attribute = "ConsumerId", Operator = "==", Value = "7" }, StartDate = DateTime.UtcNow, EndDate = DateTime.UtcNow.AddDays(7) }; activations = connector.ScheduledInstructions.Search(payload);`

Direct Consumer API call	C# code using Consumer SDK library

"CreatedBy": "SomeDomain\\Jago.Sevatarion",
"ConsumerId": 7,
"ConsumerName": "MyCustomConsumer",
"ConsumerCustomData": null,
"ParameterJson": null,
"Comments": null,
"ScheduleNextExecution": "2020-02-15T09:50:00.047Z",
"ScheduleReadableFrequency": "Occurs every 1 weeks(s) on Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday at 09:50:00 starting on 01/02/2020 ending on 16/02/2020",
"ReadablePayload": "How are network adapters configured?",
"InstructionId": null
},
{
"Id": 9,
"Name": "1E-Explorer-TachyonCore-NetworkAdapterConfigurationDetails",
"CreatedTimestampUtc": "2020-02-01T09:45:11.377Z",
"CreatedBy": "SomeDomain\\Jago.Sevatarion",
"ConsumerId": 7,
"ConsumerName": "MyCustomConsumer",
"ConsumerCustomData": null,
"ParameterJson": null,
"Comments": null,
"ScheduleNextExecution": "2020-02-16T09:50:00Z",
"ScheduleReadableFrequency": "Occurs every 1 weeks(s) on Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday at 09:50:00 starting on 01/02/2020 ending on 16/02/2020",
"ReadablePayload": "How are network adapters configured?",
"InstructionId": null
}
]
}
	"activations" object will contain the same data you can see in the JSON response on the left.

Please take note that while the JSON example has the date-time hard-coded, so it will needs adjusting when run, the C# example takes advantage of the DateTime.UtcNow to always be correct.

Entries returned in the example above have null in "InstructionId" property. This is because we've searched for entries in the future. Also, "CreatedTimestampUtc" has the timestamp of when the schedule was created, since the instruction hasn't been issued yet. If our search returned schedule activations that have already happened, the "InstructionId" would have, as name suggests, the Id of the Instruction that was issued by the schedule activation. In this case, "CreatedTimestampUtc" would have the time stamp of when the instruction was created, and not the schedule.

Let us examine the fields in the payload that we send to the API:

Property	Description
StartDate	Date time defining the start of the search window. If this is not provided the system will assume the "beginning of time" in the form of C# DateTime.MinValue, which according to Microsoft Documentation "is equivalent to 00:00:00.0000000 UTC, January 1, 0001, in the Gregorian calendar."
EndDate	EndDate must be provided if PageSize is not provided. If PageSize is provided then EndDate is optional. This property defines the end of the search window. If it is not provided the system will return enough entries to fill the PageSize provided. If it is provided the system will provide entries that fit between StartDate and EndDate only, even if there are more entries after the EndDate that would fit on the page.
PageSize	PageSize must be provided if EndDate is not provided. If EndDate is provided then PageSize is optional. This property defines how many entries should be returned. If it is not provided, the system will return all entries between StartDate and EndDate. If it is provided it will only return enough entries after the StartDate (but before EndDate, if one is provided) to fill the PageSize.
Filter	This is a filter expression that describes what schedules the system should search for. The API call will then return all activations for those schedules that are between StartDate and EndDate. This is a standard Tachyon expression and you can find more information about them on Using scope and filter expressions page.

Now we will go through a few use-cases. These will focus on the Filter property, since that if what defines which schedules will be taken into account when past activations are gathered and future activations are extrapolated.

2.5.7.3.3 Finding all schedule activations for a specific instruction definition

You can use either the Id or Name of the instruction definition to locate all schedules that will use that definition. If I wanted to look for all activations of a schedule that will issue "1E-Explorer-TachyonAgent-AgentDiagnostics" instruciton, which in my installation has the Id of 18, I would create one of these two filters:

Filter by instruction definition Id

```
{
    "Attribute": "DefinitionId",
```

```
    "Operator": "==",
    "Value": "18"
}
```

or

Filter by instruction definition Name

```
{
    "Attribute": "DefinitionName",
    "Operator": "==",
    "Value": 1E-Explorer-TachyonAgent-AgentDiagnostics
}
```

2.5.7.3.4 Finding all schedule activations for a specific scheduled instruction

When you create a scheduled instruction (see below), you receive its Id. You can then use that Id to track instructions issued by the schedule.

Find all instructions issued for a specific schedule

```
{
    "Attribute": "ScheduledInstructionId",
    "Operator": "==",
    "Value": "1"
}
```

2.5.7.3.5 Finding all schedule activations that contain a marker

You can use ConsumerCustomData property in the payload sent to the API when you create a scheduled instruction to create a "marker", which you can then use to find scheduled instructions. This field is not examined or used by Tachyon, but it will be copied from the Schedule into each instance of an instruction issued by said Schedule.

Find all instructions that bear a specific marker

```
{
    "Attribute": "ConsumerCustomData",
    "Operator": "==",
    "Value": "My special instruction marker"
}
```

2.5.7.3.6 Complex example

Lastly, let's look at the entire payload (so not just the filter) for one last use case. Let us assume that various consumers schedule instructions and when those instructions are issued they are given a marker which defines what Organizational Unit they are issued by. Now let's assume we would like to find all scheduled activations of instruction definition "1E-Explorer-TachyonCore-DeviceDrivers" in May 2020 for consumer "MyCustonConsumer", which has the Id of 7 that had the marker value of "OU:Finance".

Request payload Expand source

```
{
    "Filter": {
        "Operator": "AND",
        "Operands": [
            {
                "Attribute": "ConsumerCustomData",
                "Operator": "==",
                "Value": "OU:Finance"
            },
            {
                "Attribute": "ConsumerId",
                "Operator": "==",
                "Value": "7"
            },
            {
                "Attribute": "DefinitionName",
                "Operator": "==",
                "Value": "1E-Explorer-TachyonCore-DeviceDrivers"
            }
        ]
    },
    "StartDate": "2020-05-01T00:00:00.001Z",
    "EndDate": "2020-05-31T23:59:59.999Z"
}
```

2.5.7.4 Lifecycle of a schedule

Lifecycle of a schedule is similar to that of an instruction. By default, Tachyon is configured to require authentication and approval for each schedule. Schedule will be "live" only once creating user has authenticated themselves and the schedule has been approved, providing is has been created with the ScheduleEnabled flag set to true.

A schedule will remain live between its starting and finishing dates, so between "ScheduleActiveStartDate" and "ScheduleActiveEndDate". How the schedule is triggered/activated between those dates depends on its settings. Once the schedule finishes, its Status will be set to Expired (7).

If the user creating a schedule fails to authenticate is good time, the schedule will fail. If the user authenticates, the schedule then needs to be approved and a schedule will remain in "InApproval" (1) state until it is either rejected or approved, so potentially can be stuck for a long time.

The requirement to authenticate the user creating a schedule isn't ideal if these schedules are created programmaticaly.

To alleviate that, you can either turn off 2FA completely, which is not advised or just for scheduled instructions, which is the preferred method.

To configure Tachyon not to require authentication for scheduled instructions open Tachyon Master database and locate a table called "GlobalSettings". In that table there will be a row with the name of "DefaultWorkflow". You have to examine the Value and change it.

By default, it looks like this:

DefaultWorkflow row value as configured by Tachyon installation Expand source

```
[
    {
        "ReferenceType": 0,
        "InstructionWorkflow": [
            {
                "InstructionType": 0,
                "Workflow": {
                    "StateMachine": "State"
                }
            },
            {
                "InstructionType": 1,
                "Workflow": {
                    "StateMachine": "State with Authentication and Approval"
                }
            }
        ]
    },
    {
        "ReferenceType": 2,
        "Workflow": {
            "StateMachine": "Scheduled Instruction State with Authentication an
d Approval",
            "Authenticating": "Scheduled Instruction Authentication",
            "Approval": "Scheduled Instruction Approval",
            "InProgress": "Scheduled Instruction InProgress",
            "Cancelling": "Scheduled Instruction Cancelling"
        }
    },
    {
        "ReferenceType": 3,
        "Workflow": {
            "StateMachine": "Agent Deployment State",
            "Processing": "Agent Deployment Processing",
            "Completing": "Agent Deployment Completing",
            "Cancelling": "Agent Deployment Cancelling"
        }
    },
    {
        "ReferenceType": 4,
        "Workflow": {
            "StateMachine": "Management Group Sync State",
            "Processing": "Management Group Sync Processing",
            "Completing": "Management Group Sync Completing"
        }
    },
    {
        "ReferenceType": 5,
        "Workflow": {
            "StateMachine": "Policy Deployment State",
```

```
            "Processing": "Policy Deployment Processing",
            "Completing": "Policy Deployment Completing"
        }
    }
]
```

Locate the JSON entry that reads:

"StateMachine": "Scheduled Instruction State with Authentication and Approval"

and change it to:

"StateMachine": "Scheduled Instruction State with Approval".

Save the modified value into the database.

2.5.7.5 Create a new schedule

In this section we will learn how to create a schedule.

When designig your schedule you have to take into account that a schedule can have two intervals. The main one, described by "ScheduleFreqType" and "ScheduleFreqInterval" can be supplemented by an "inner" one, each one occuring within a single "tick" of the main interval. This "inner" interval is described by "ScheduleFreqSubdayType" and ""ScheduleFreqSubdayInterval" properties.

This means that in some cases, there are multiple ways of creating effectively the same schedule. For example, if you wanted to create a schedule that executes daily at a given time, you could define a schedule, where the main schedule is set to daily and the inner one defiens a single execution at a given point in time, but you could also define a schedule where the main schedule is set to weekly and the inner one is set to executing once a day on every day of the week.

The "daily" version would look like this:

Schedule that executes an instruction once a day - "daily" version Expand source

```
{
    "Comments": null,
    "DefinitionName": "1E-Explorer-TachyonCore-AllInstalledSoftware",
    "InstructionTtlMinutes": 1440,
    "KeepRaw": 1,
    "Parameters": null,
    "ResponseTtlMinutes": 1440,
    "ScheduleActiveStartDate": "2020-2-15T00:00:00Z",
    "ScheduleActiveStartTime": 500,
    "ScheduleActiveEndDate": "2025-2-12T00:00:00Z",
    "ScheduleActiveEndTime": null,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": null,
    "ScheduleFreqRelativeInterval": null,
    "ScheduleFreqRecurrenceFactor": null,
}
```

While the "weekly" version would look like this:

Schedule that executes an instruction once a day - "weekly" version Expand source

```
{
    "Comments": null,
    "DefinitionName": "1E-Explorer-TachyonCore-AllInstalledSoftware",
    "InstructionTtlMinutes": 1440,
    "KeepRaw": 1,
    "Parameters": null,
    "ResponseTtlMinutes": 1440,
    "ScheduleActiveStartDate": "2020-2-15T00:00:00Z",
    "ScheduleActiveStartTime": 500,
    "ScheduleActiveEndDate": "2025-2-12T00:00:00Z",
    "ScheduleActiveEndTime": 235900,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 8,
    "ScheduleFreqInterval": 127,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": 0,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 1,
}
```

Finally, let's look at which API endpoint and SDK method to use when creating a schedule. I'll pick the "daily" one for the example.

Direct Consumer API call	C# code using Consumer SDK library
By making a POST request to https://my.tachyon.server/Consumer/ScheduledInstructions you can schedule an instruction using instruction definition's name: **Request payload** Expand source `{ "Comments": null, "DefinitionName": "1E-Explorer-TachyonCore-AllInstalledSoftware", "InstructionTtlMinutes": 1440, "KeepRaw": 1, "Parameters": null, "ResponseTtlMinutes": 1440, "ScheduleActiveStartDate": "2020-2-15T00:00:00Z", "ScheduleActiveStartTime": 500, "ScheduleActiveEndDate": "2025-2-12T00:00:00Z", "ScheduleActiveEndTime": null,`	Use ScheduledInstructions object inside the Tachyon connector instance. **Scheduling an instruction to run daily** Expand source `var payload = new ScheduledInstruction { DefinitionName = "1E-Explorer-TachyonCore-AllInstalledSoftware", InstructionTtlMinutes = 1440, KeepRaw = true, Parameters = null, ResponseTtlMinutes = 1440, ScheduleActiveStartDate = new DateTime(2020, 2, 15, 0,`

Direct Consumer API call

```
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": null,
    "ScheduleFreqRelativeInterval": null,
    "ScheduleFreqRecurrenceFactor": null,
}
```

or by using instruction definition's Id:

Request payload Expand source

```
{
    "Comments": null,
    "DefinitionId": 44,
    "InstructionTtlMinutes": 1440,
    "KeepRaw": 1,
    "Parameters": null,
    "ResponseTtlMinutes": 1440,
    "ScheduleActiveStartDate": "2020-2-15T00:0
0:00Z",
    "ScheduleActiveStartTime": 500,
    "ScheduleActiveEndDate": "2025-2-12T00:00:
00Z",
    "ScheduleActiveEndTime": null,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": null,
    "ScheduleFreqRelativeInterval": null,
    "ScheduleFreqRecurrenceFactor": null,
}
```

The return payload will contain information about the instruction we just scheduled:

Return payload Expand source

```
{
    "Id": 9,
    "Name": "1E-Explorer-TachyonCore-AllInstal
ledSoftware",
    "Cmd": "SendAll",
    "KeepRaw": true,
    "Scope": null,
    "ReadableScope": null,
    "InstructionTtlMinutes": 1440,
```

C# code using Consumer SDK library

```
0, 0),
    ScheduleActiveStartTime =
500,
    ScheduleActiveEndDate = n
ew DateTime(2025, 2, 12, 0,
0, 0),
    ScheduleActiveEndTime = n
ull,
    ScheduleEnabled = true,
    ScheduleFreqType = 4,
    ScheduleFreqInterval = 1,
    ScheduleFreqSubdayType =
1,
    ScheduleFreqSubdayInterva
l = null,
    ScheduleFreqRelativeInter
val = null,
    ScheduleFreqRecurrenceFac
tor = null,
};
var scheduledInstruction = co
nnector.ScheduledInstruction
s.SendScheduledInstruction(pa
yload);
```

"scheduledInstruction" object will contain the same data you can see in the JSON response on the left.

```
    "ResponseTtlMinutes": 1440,
    "CreatedTimestampUtc": "2020-02-12T08:37:3
6.9Z",
    "Export": false,
    "ExportLocation": null,
    "ParentInstructionId": null,
    "InstructionDefinitionId": 36,
    "CreatedBy": "SomeDomain\\Marius.Gage",
    "ResultsFilter": null,
    "PreviousResultsFilter": null,
    "Devices": null,
    "ConsumerId": 1,
    "ConsumerName": "MyCustomConsumer",
    "ConsumerCustomData": null,
    "ParameterJson": null,
    "OffloadResponses": false,
    "RequestedFor": null,
    "Comments": null,
    "ActionedBy": null,
    "ActionReason": null,
    "Status": 0,
    "StatusTimestampUtc": null,
    "Workflow": "{\"StateMachine\":\"Scheduled
Instruction State with Authentication and Appr
oval\",\"Authenticating\":\"Scheduled Instruct
ion Authentication\",\"Approval\":\"Scheduled
Instruction Approval\",\"InProgress\":\"Schedu
led Instruction InProgress\",\"Cancelling\":\"
Scheduled Instruction Cancelling\"}",
    "WorkflowState": 0,
    "ApprovalOffloaded": false,
    "ReadablePayload": null,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": 0,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 0,
    "ScheduleActiveStartDate": "2020-02-15T00:
00:00Z",
    "ScheduleActiveEndDate": "2025-02-12T00:0
0:00Z",
    "ScheduleActiveStartTime": 500,
    "ScheduleActiveEndTime": null,
    "ScheduleLastExecuted": null,
    "ScheduleNextExecution": "2020-02-15T00:0
5:00Z",
```

```
    "ScheduleReadableFrequency": "Occurs every
1 day(s) at 00:05:00 starting on 15/02/2020 en
ding on 12/02/2025"
}
```

2.5.7.5.1 ScheduleActiveStartTime and ScheduleActiveEndTime

These two properties for the in-day boundaries of a schedule and their value will have different meaning depending on how the schedule is set up.

If the "main" schedule is defined as a daily schedule, then you can define the number of repeats each day. Those repeats will happen between **ScheduleActiveStartTime** and **ScheduleActiveEndTime** time.

If the "main" schedule is not a daily one, then **ScheduleActiveStartTime** will define the time of day on which the schedule should be activated.

2.5.7.5.2 Narrowing the set of devices that should respond to the scheduled instruction

Much like with regular instructions, scheduled ones can be scoped and targeted. The scope expression or targer list will be applied to each instruction issued on the schedule.

Issue an instruction, track its progress and retrieve responses describes how scoping and targeting works for instructions. To add scope you'd simply add **Scope** property to the schedule creating payload and you would build an expression defining a scope, exactly like you would for regular instruction.

For targeted schedule instructions, you would include the Devices array in the payload and instead of POSTing the payload to https://my.tachyon.server/Consumer/ScheduledInstructions you would POST to https://my.tachyon.server/Consumer/ScheduledInstructions/Targeted. If you're using the Consumer SDK, you would call connector.ScheduledInstructions.SendTargetedScheduledInstruction() instead of connector.ScheduledInstructions.SendScheduledInstruction().

2.5.7.5.3 Other instruction properties

As you have seen on Issue an instruction, track its progress and retrieve responses, instructions have other properties, like Parameters, ResultsFilters, Export options etc. These properties are the same on the scheduled instructions and what you set for the schedule will be copied into each instance of instruction issued by the schedule.

2.5.7.6 Modifying and cancelling a schedule

An existing schedule can be modified by using a PUT verb, instead or POST while calling either https://my.tachyon.server/Consumer/ScheduledInstructions or https://my.tachyon.server/Consumer/ScheduledInstructions/Targeted. You have to include the schedule Id (in the example above the schedule we created had an Id of 9) in the payload as well. If you're using the Consumer SDK, instead or using connector.ScheduledInstructions.SendTargetedScheduledInstruction() and connector.ScheduledInstructions.SendScheduledInstruction() you would use

connector.ScheduledInstructions.UpdateScheduledInstruction() and connector.ScheduledInstructions.UpdateTargetedScheduledInstruction() respectively:

Direct Consumer API call	C# code using Consumer SDK library
PUT request to https://my.tachyon.server/Consumer/ScheduledInstructions: **Request payload** Expand source `{ "Id": 9, "Comments": null, "DefinitionName": "1E-Explorer-TachyonCore-AllInstalledSoftware", "InstructionTtlMinutes": 1440, "KeepRaw": 1, "Parameters": null, "ResponseTtlMinutes": 1440, "ScheduleActiveStartDate": "2020-2-15T00:00:00Z", "ScheduleActiveStartTime": 500, "ScheduleActiveEndDate": "2025-2-12T00:00:00Z", "ScheduleActiveEndTime": null, "ScheduleEnabled": true, "ScheduleFreqType": 4, "ScheduleFreqInterval": 1, "ScheduleFreqSubdayType": 1, "ScheduleFreqSubdayInterval": null, "ScheduleFreqRelativeInterval": null, "ScheduleFreqRecurrenceFactor": null, }` will yield result we've already seen when creating the schedule	Use ScheduledInstructions object inside the Tachyon connector instance. **Updating a schedule** Expand source `var payload = new ScheduledInstruction { DefinitionName = "1E-Explorer-TachyonCore-AllInstalledSoftware", InstructionTtlMinutes = 1440, KeepRaw = true, Parameters = null, ResponseTtlMinutes = 1440, ScheduleActiveStartDate = new DateTime(2020, 2, 15, 0, 0, 0), ScheduleActiveStartTime = 500, ScheduleActiveEndDate = new DateTime(2025, 2, 12, 0, 0, 0), ScheduleActiveEndTime = null, ScheduleEnabled = true, ScheduleFreqType = 4, ScheduleFreqInterval = 1, ScheduleFreqSubdayType = 1, ScheduleFreqSubdayInterval = null, ScheduleFreqRelativeInterval = null, ScheduleFreqRecurrenceFactor = null, };`

Direct Consumer API call	C# code using Consumer SDK library
	`var scheduledInstruction = connector.ScheduledInstructions.UpdateScheduledInstruction(9, payload);` "scheduledInstruction" object will contain the same data we've seen in the example where we created the schedule

To cancel a schedule all you need is its Id

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/ScheduledInstructions/9 will result in the schedule with the Id of 9 being deleted.	Use ScheduledInstructions object inside the Tachyon connector instance. **Cancelling a schedule** Expand source `connector.ScheduledInstructions.CancelSchedule(9);`

2.5.7.7 Example schedules

In this section we will look at several schedules. We will focus on the scheduling properties of the payload for brevity and assume each schedule issues the same instruction. Please remember that all times are in UTC.

First let's look at a schedule that executes just once, at 19:30 on 18th of February 2020.

Execute once schedule Expand source

```
{
   "ScheduleActiveStartDate": "2020-2-18T00:00:00Z",
   "ScheduleActiveStartTime": 193000,
   "ScheduleActiveEndDate": null,
   "ScheduleActiveEndTime": null,
   "ScheduleEnabled": true,
   "ScheduleFreqType": 1,
   "ScheduleFreqInterval": 0,
```

```
   "ScheduleFreqSubdayType": 0,
   "ScheduleFreqSubdayInterval": 0,
   "ScheduleFreqRelativeInterval": 0,
   "ScheduleFreqRecurrenceFactor": 0
}
```

Now let's look at a more complicated use case. Imagine if you will that you have a task that requires user interaction and should be completed within a week. Although it requires user interaction, you can still check if it has been done via an instruction, and you want to get the notification that a task has been done within the hour if possible. You decide to to create a schedule where the check instruction is executed every day, repeated every hour within business hours (9 to 17), as there is no point in checking when there's no one in the office.

Execute once a day for a week, 8 times a day Expand source

```
{
    "ScheduleActiveStartDate": "2020-2-22T00:00:00Z",
    "ScheduleActiveStartTime": 90000,
    "ScheduleActiveEndDate": "2020-2-29T00:00:00Z",
    "ScheduleActiveEndTime": 170000,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 4,
    "ScheduleFreqInterval": 1,
    "ScheduleFreqSubdayType": 8,
    "ScheduleFreqSubdayInterval": 1,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 0
}
```

This time let's say you'd like to, on a Friday evening (18:00) run an instruction that, for instance, checks what new software was installed in the last 7 days. To do that you could create a weekly schedule that executes only on Friday at 18:00:

Execute once a week on Friday at 18:00 Expand source

```
{
    "ScheduleActiveStartDate": "2020-2-22T00:00:00Z",
    "ScheduleActiveStartTime": 180000,
    "ScheduleActiveEndDate": "2025-2-15T00:00:00Z",
    "ScheduleActiveEndTime": 235900,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 8,
    "ScheduleFreqInterval": 32,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": 0,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 1
}
```

Lastly, let's look at a schedule that runs an instruction once a month, on the 7th day of the month at 9:00.

Execute once a month on the 7th day of the month at 9:00 Expand source

```
{
    "ScheduleActiveStartDate": "2020-2-24T00:00:00Z",
    "ScheduleActiveStartTime": 90000,
    "ScheduleActiveEndDate": "2025-2-24T00:00:00Z",
    "ScheduleActiveEndTime": null,
    "ScheduleEnabled": true,
    "ScheduleFreqType": 16,
    "ScheduleFreqInterval": 7,
    "ScheduleFreqSubdayType": 1,
    "ScheduleFreqSubdayInterval": 0,
    "ScheduleFreqRelativeInterval": 0,
    "ScheduleFreqRecurrenceFactor": 1
}
```

2.5.8 Set up Principals, Roles and Permissions, for Tachyon version 8.0 and later

This section contains an overview of Tachyon role based access control (RBAC) and its components and how to configure Tachyon RBAC. This is not intended as an in-depth explanation of what RBAC is, but as a demonstration of how it's been implemented in Tachyon and how users can configure it programmatically.

RBAC implementation in Tachyon has had a major rework in version 8.0. This page will highlight differences between 'old' and 'new' implementations.

The C# examples assume you're using Tachyon Consumer SDK and you have an instantiated an instance of Tachyon connector class in an object called connector.All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API.

In the following examples this detailis leftout, stating that the returned object contains the data.For example, when we say that XYZ object contains certain data, this means the ReceivedObject property inside said object contains that data.

2.5.8.1 Basics of role based access control (RBAC)

Role-based access control is an access control mechanism defined around roles and privileges. This security model pivots around the concept or a Role. Users (called principals in Tachyon) can be assigned to a Role and it's through a Role they gain permissions to perform actions. Each element of Tachyon's security system leads back to a Role.

In Tachyon 8.0, a third element has been added - a Management Group. How it comes into play you can read further down.

2.5.8.2 RBAC objects in Tachyon

Tachyon has several types of objects that are part of its RBAC system. This section serves as a brief description of these objects and their purpose.

2.5.8.2.1 Principals

Principals are identical to users. They're not called users because the word 'user' is traditionally associated with a person. A principal is, from a technical standpoint, an Active Directory account which may be a user account or a computer account. It can also be an AD group.

Tachyon allows access only to principals authenticated through Windows authentication and known to Tachyon itself. Depending on the request, Tachyon establishes the permissions of the calling principal by looking at the roles that principal is assigned to. By default, a fresh installation of Tachyon has a principal representing the user account installing it and another principal for the "NT AUTHORITY\Network Service", which many of Tachyon's services will be running as.

2.5.8.2.2 Roles

A Role is a container for Permissions and all permissions are linked to roles.

Principals must have roles assigned to them before using the system.

2.5.8.2.3 Permissions

A permission is an ability to perform an operation on a securable type.

2.5.8.2.4 Securable Types

A Securable Type represents a type of an object that can have permissions assigned to it. For example, Instructions can have permissions, as can Consumers and Management Groups. Security itself is an object principals need permissions to, so they can modify it, it will also have a Securable Type. If an element of Tachyon has security defined for it, it must have a securable type.

Instances

An Instance is one, specifc copy of an object of a given Securable Type. This can, for instance, be a single Instruction Set.

While most permissions work on Securable Types as a general thing, where if a Principal is assigned a Permission on a given Securable Type they can work with any object of that type, some Securable Types allow Permissions to be specified on just one given instance of a Securable Type object, giving a more granular control over access to that Securable Type.

At the moment only InstructionSet securable type supports instances.

2.5.8.2.5 Operations

An Operation (also called Applicable Operation) represents the type of an action that can be performed on a securable type, like "Read" or "Write".

2.5.8.2.6 Management Group

A Management Group is a container for Devices. A Management Group can either define rules, which are then used to decide which Devices belongs to a given group, or can explicitly list Devices that should belong to the Management Group.

While Management Groups are discussed in this article, they are not a focus of it and the nature of their creation and evaluation is covered elsewhere.

A Management Group and contain other Management Groups, with child Management Groups inheriting all rules of the parent and adding new ones, narrowing the device selection further with each level of nesting.

Permissions are inherited throughout the Management Group structure. This means that for RBAC purposes having Permissions on a Management Group means a automatically having the same Permissions on any and all children of that Management Groups.

2.5.8.2.7 Example how to read permissions

Marc (**Principal**) through his All Instructions Questioners (**Role**) has Questioner (**Applicable Operation**) permission on Instructions (**Securable type**) assigned on USA-Engineering (**Management Group**).

A Principal will have a specific permission on given securable type through a role the principal belongs to. This is the only way principals can obtain permissions and permissions are always for a specific operation on a given type.

To establish the full permission set of a given principal you have to combine all permissions from all the roles assigned to that principal.

2.5.8.3 Changes introduced in Tachyon version 8.0

There are two main changes introduced in version 8.0. The first is the very mechanism by which a User is assigned to a Role, and the second is the concept of delegation. There are, however, other, auxiliary concepts and changes that we will cover in this section.

2.5.8.3.1 Principals, Roles and Management Groups

In previous versions of Tachyon a Principal was assigned to a Role.

Furthermore, a Role could be linked to a Management Group, providing said Role had an Instruction Set related permission(s).

That way, a Role could be linked to given Management Group(s), and that link would be passed on to every Principal assigned to the Role. That way, a Role could be created with access to a specific Management Group(s) before any Principals have been assigned to the Role.

In version 8.0, Management Groups cannot be linked directly to Roles. Instead, RBAC assignments follow a "Who, What, Where" pattern - "Who" can perform "What" action(s) "Where".

"Who" is a Principal, so a User or a Group that performs various actions in the system.

"What" is the set of Permissions assigned to a Role. We will be talking about a Role, as it is little more than a container for Permissions, but we should remember that it is the Permissions assigned to a Role that defined what the Role's assignee can do.

"Where" is the collection of Devices on which the Principal can excercise their Permissions. These collections are Management Groups.

This means that from version 8.0 onwards, Principals are assigned to a Role on a specific Management Group, so given Principal will be able to excercise Pemrissions of the Role they've been assigned on a given Management Group and not elsewhere.

As an example, when a Principal wished to excercise their "Actioner" Permission they have via 'All Instructions Actioner' Role, they will only be able to do so on a specific Management Group the is defined in the assignment.

2.5.8.3.2 Global and Local Permissions

Not all Securable Types are connected to Management Groups. Many types are used to secure parts of the system not connected in any way with Devices, hence they are ignorant of Management Groups.

These Securable Types are called 'global' securable types and are easily identifiable as they have 'IsGlobal' flag set to true.

Because global Securable Types have no link to Devices or Management Groups, it is irrelevant what Management Group is assigned to Role that uses global securable types in its Permissions.

A Securable Type that has the 'IsGlobal' flag set to false is called a Local type and Permissions that use that Securable Type are called Local or Localized Permissions.

This means that the Management Group part of the assignment will define the boundaries of where the Permission can be excercised.

2.5.8.3.3 Role types

Because some Permissions are localized and some are global, depending on which Securable Type they use, we now have 3 types of Roles: Global, Local and Hybrid.

- Global Roles - these roles contain only permissions that use SecurableTypes which have 'IsGlobal' flag set tot true.
- Local Roles - these roles contain only permissions that use Securable Types which have 'IsGlobal' flag set to false.
- Hybrid Roles - these roles contain a mixture of permissions that use both global and non-global Securable Types.

A Local Role can be maked as delegatable. Global and Hybrid roles are not delegatable.

Non-delegatable Roles (be that Global, Hybrid or Local) can only be assigned to "All Devices", with the exception of "Group Administrator" Role, which cannot be assigned to All Devices despite being a hybrid Role.

Delegatable Roles, except for "Group Administrator", can be assigned to any Management Group, including "All Devices", subject to caller's Security Permissions.

You also have to remember that only Local Roles can be made delegatable. You can read about delegation here.

2.5.8.3.4 Anatomy of an RBAC assignment

An Assignment is a three-way "who, what, where" relationship between a Principal, Role and a Management Group.

A single Assignment is between a single Principal, a single Role and a single Management Group.

If you wish for a Principal to be able to exercise given Role's Permissions on two unrelated (see here) Management Groups, you will need to create two assignments.

You can create any number of assignments as long as they are all for unique combination of Principal, Role and Management Group.

2.5.8.3.5 Delegation

Delegation is, in broadest terms, a concept where an administrator would delegate some of their own responsibility to other users while limiting the area where they can excercise that responsibility.

While the concept itself is outside the bounds of this page, we will cover how delegation works internally in Tachyon and what implications it has to API users.

2.5.8.3.5.1 Types of security administrator

Before we delve into how the feature works, we have to establish two terms that we will use later on: global security administrator and local security administrator.

A global security administrator is a Principal that has a Role with a Permission that uses "Security" Securable Type and that Role is assigned to "All Devices" Management Group.

A local security administrator is a Principal that has a Role with a Permission that uses "Security" Securable Type and that Role is assigned on a Management Group other than "All Devices".

It is important to understand that global security administrator and local security administrator are not explicit Roles. Any Role with "Security" Permissions can make a Principal be considered global or local security administrator, depending on what Management Group said Role is assigned on.

One should also remember that the global and local security administrator aren't actually referenced in Tachyon and are not terms used by documentation in general. We define them here because they will be used on this page to descirbe how delegation works.

2.5.8.3.5.2 Who can assign what?

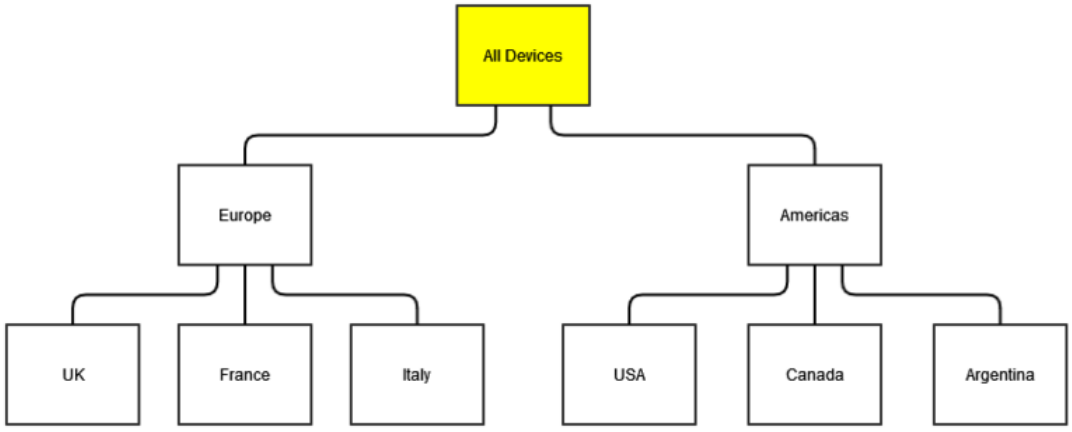
A global security administrator can create and modify Roles. They can also assign any Role in the system to any Principal using any Management Group they see fit (with some exceptions, like the fact that "Group Administrator" Role cannot be assigned to "All Devices" at all).

A local security administrator cannot create or modify Roles. They can assign any Role that is marked as delegatable ("CanBeDelegated" flag is set to true) to a Management Group they have the Security Permission on or any of its children. The exception here is that a local security administrator cannot assign a Role that has "Security" Permission using the same Management Group they themselves have "Security" Permission on. They have to use a child Management Group of the one they themselves have "Security" Permission on.

Effectively, a local security administrator cannot create an assignment that would result in another Principal having the same "Security" Permissions as they do. This limitation does not apply to global security administrator.

2.5.8.3.5.3 An example of how it works

Let's assume we have following Management Group structure:



We also have a Role called 'Security Administrator', which has Permission on Security (Read and Write), and another role called "Actioner" which has Actioner permission on an Instruction Set.

Now let us assume we have 3 Principals. First Principal, called 'John', has the 'Security Administrator' Role assigned on 'All Devices' Management Group.

The second Principal, called 'Jane', has, at the moment, no Roles assigned at all.

The third Principal, called 'Frank", has no Roles assigned either.

At this point John, as per the rules outlined above, can create and modify Roles and assign any Role to any Principal using any Management Group.

John decides that he is going to delegate some of his responsibilities to Jane by assigning her to 'Security Administrator' on 'Europe' Management Group.

This means Jane is not able to create or edit Roles because, as per rules outlined above, she is a local security administrator. When manging assignments she can also only use "Europe", "UK", "France" and "Italy" Management Groups, because she has Security permission assigned on "Europe" and "UK", "France" and "Italy" are child Management Groups of "Europe". She is also limited to using only Roles that are marked as 'delegatable'.

Jane can assign "Actioner" role to Frank on "Europe" Management Group, because that Role does not have Security Permission.

Jane will not be able to assign "Security Administrator" Role to Frank using "Europe" Management Group, because she herself has Security Permission on that very Management Group and because that Role does have a Security Permission. She can, however, assign "Security Administrator" to Frank using either "UK", "France" or "Italy" Management Groups, because those are children Management Groups or the Management Group she has Security Permission on.

2.5.8.4 Retrieving RBAC objects

In this section we will look at endpoints that allow you to examine existing RBAC objects.

2.5.8.4.1 Getting Principals

The most basic functionality is to retrieve all Principals:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals will yield following response: **Return payload** Expand source [{ "Id": 1, "ExternalId": "S-1-5-21-1202660629-789336158-1349024091-27850", "PrincipalName": "SomeDomain\\Administrator", "Email": "Administrator@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": true, "DisplayName": "Administrator", "IsGroup": false }, { "Id": 2, "ExternalId": "S-1-5-20", "PrincipalName": "NT AUTHORITY\\Network Service", "Email": null, "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:56.687Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.687Z", "SystemPrincipal": true, "DisplayName": "Network Service", "IsGroup": false }, { "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true,	Use Principals object inside the Tachyon connector instance. **Retrieving all principals** Expand source `principals = onnector.Principals.GetAll();` "principals" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library


```
        "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
        "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
        "SystemPrincipal": false,
        "DisplayName": "Jane Doe",
        "IsGroup": false
    },
    {
        "Id": 4,
        "ExternalId": "S-1-5-21-3276326578-728
399001-2836074973-1009",
        "PrincipalName": "SomeDomain\\John.Doe
",
        "Email": "John.Doe@SomeDomain",
        "Enabled": true,
        "CreatedTimestampUtc": "2016-11-30T15:
10:10.73Z",
        "ModifiedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
]
```


Or just a single Principal by their Id. Here we'll look for "SomeDomain\\Jane.Doe", who in has the Id of 3:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals/3 will yield following response: **Return payload** Expand source `{ "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z",`	Use Principals object inside the Tachyon connector instance. **Retrieving specific Principal** Expand source `principal = connector.Principals.Get(3);` "principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "SystemPrincipal": false,
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```

You can also retrieve all Principals that have a specific Role assigned to them. Here we'll get all Principals who are "Global Administrators", which has the Id of 1:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals/Role/1 will yield following response: **Return payload** Expand source	Use Principals object inside the Tachyon connector instance. **Retrieving all principals that have a specific role** Expand source `principals = connector.Principals.GetForRole(1);` "principals" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "PrincipalId": 1,
        "RoleId": 1,
        "CreatedTimestampUtc": "2019-11-
07T13:15:01.533Z",
        "Role": null,
        "Principal": {
            "Id": 1,
            "ExternalId": "S-1-5-21-1202
660629-789336158-1349024091-27850",
            "PrincipalName": "SomeDomain
\\Administrator",
            "Email": "Administrator@Some
Domain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019
-11-07T13:14:52.777Z",
            "ModifiedTimestampUtc": "201
9-11-07T13:14:52.777Z",
            "SystemPrincipal": true,
            "DisplayName": "Administrato
r",
            "IsGroup": false
        }
    }
]
```

2.5.8.4.2 Getting Roles

Much like with Principals, you can retrieve all Roles in the system:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles will yield following response **Return payload** Expand source	Use Roles object inside the Tachyon connector instance. **Retrieving all Roles** Expand source `roles = connector.Roles.GetAll();` "roles" object will contain the same data you can see in the JSON response on the left.


```
[
    {
        "Id": 27,
        "Name": "1E ITSM Connect Actioner",
        "Description": "The ServiceNow proxy user is added to this role instead of Global Actioners so that ServiceNow users can only use instructions belonging to instruction sets assigned to this role",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.51Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": false,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 7,
        "Name": "All Instructions Actioner",
        "Description": "This role will allow assigned users to execute all instructions that are defined as Actions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.063Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": false
    },
    {
        "Id": 5,
        "Name": "All Instructions Approver",
        "Description": "This role will allow assigned users to approve all instructions that require approval.",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.84Z",
```

```
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 6,
        "Name": "All Instructions Questioner",
        "Description": "This role will allow assigned
users to execute all instructions that are defined as
Questions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.06
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 8,
        "Name": "All Instructions Viewer",
        "Description": "This role will allow assigned
users to view responses to all instructions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.06
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 40,
        "Name": "Area Administrator",
        "Description": "The Area Administrator role al
lows users assigned to assign users permission that ha
ve been defined as delegatable in Tachyon, this means
that they can be restricted to a Management Group.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.21
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.2
13Z",
```

```
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": true
    },
    {
        "Id": 32,
        "Name": "Engagement Administrator",
        "Description": "View, create, update, delete a
nd enable Engagements",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.01
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.0
1Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": false
    },
    {
        "Id": 41,
        "Name": "Experience Administrator",
        "Description": "The Experience Administrator r
ole allows users assigned to access all areas of Exper
ience, including defining Engagement Surveys and Annou
ncements.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.22
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.2
2Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 38,
        "Name": "Experience Engagement Assigner",
        "Description": "This role will allow assigned
users the ability to assign Engagements to any Managem
ent Group that is including in their assignments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.97
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.9
7Z",
        "SystemRole": false,
        "CanBeDelegated": true,
```

```
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 30,
        "Name": "Experience User",
        "Description": "The Experience User role allow
s users assigned to view access all areas of Experienc
e.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.83
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 1,
        "Name": "Full Administrator",
        "Description": "The Full Administrator has the
combined permissions of all other roles.",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.83
7Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": true
    },
    {
        "Id": 24,
        "Name": "Guaranteed State Administrator",
        "Description": "The Guaranteed State Administr
ator role allows users assigned to access all areas of
Guaranteed State including defining Rules, Polices and
uploading fragments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.46
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
```

```
    },
    {
        "Id": 37,
        "Name": "Guaranteed State Policy Assigner",
        "Description": "This role will allow assigned
users the ability to assign Policies to any Management
Group that is including in their assignments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.96
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.9
6Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 25,
        "Name": "Guaranteed State User",
        "Description": "The Guaranteed State User role
allows users assigned to view access all areas of Guar
anteed State.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.46
7Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 39,
        "Name": "Installer",
        "Description": "The Installer role has minimum
permissions to install the Platform and Applications,
register Consumers, upload Products packs and manage I
nstructions sets.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.20
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.2
03Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": true
    },
```

```
    {
        "Id": 13,
        "Name": "Management Group Administrator",
        "Description": "Create, delete, update and ini
tiate synchronization of Management Groups",
        "CreatedTimestampUtc": "2021-10-07T10:51:36.98
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 44,
        "Name": "Nomad Administrators",
        "Description": null,
        "CreatedTimestampUtc": "2021-10-07T13:42:13.52
3Z",
        "ModifiedTimestampUtc": "2021-10-07T13:42:13.5
23Z",
        "SystemRole": false,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 42,
        "Name": "Patch Success Administrator",
        "Description": "The Patch Success Administrato
r role allows users assigned to access all areas of Pa
tch Success including deploying Patches.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.22
3Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.2
23Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 26,
        "Name": "Patch Success User",
        "Description": "The Patch Success User role al
lows users assigned to view access all areas of Patch
```

Direct Consumer API call	C# code using Consumer SDK library

```
Success and initiate check status and update status.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.49
Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.8
93Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 43,
        "Name": "Tachyon System",
        "Description": "The System role allows assigne
d users to performance operations required to execute
the different cross component activities needed for Ta
chyon Platform to run normally.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.22
7Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.2
27Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": true
    }
]
```

But unlike the Principals, roles also have an endpoint that supports filtering, sorting and paging. In the example below we'll look for custom roles and sort them by name.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Search with following payload:	Use Roles object inside the Tachyon connector instance.
Request payload Expand source	**Searching for Roles** Expand source

```
{
 "PageSize": 10,
 "Start": 1,
 "Filter": {
 "Attribute": "SystemRole",
 "Operator": "==",
```

```
var payload = new Se
arch
{
        PageSize = 1
0,
```

Direct Consumer API call

```
 "Value": "false"
 },
 "Sort": [{
 "Direction": "ASC",
 "Column": "Name"
 }]
}
```

Yields following response:

Return payload Expand source

```
{
    "TotalCount": 7,
    "Items": [
        {
            "Id": 27,
            "Name": "1E ITSM Connect Actioner",
            "Description": "The ServiceNow proxy user i
s added to this role instead of Global Actioners so tha
t ServiceNow users can only use instructions belonging
to instruction sets assigned to this role",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
7.51Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
7.893Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 41,
            "Name": "Experience Administrator",
            "Description": "The Experience Administrato
r role allows users assigned to access all areas of Exp
erience, including defining Engagement Surveys and Anno
uncements.",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
9.22Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
9.22Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
```

C# code using Consumer SDK library

```
        Start = 1,
        Filter = new
ExpressionObject
        {
                Attr
ibute = "SystemRole
",
                Oper
ator = "==",
                Valu
e = "false"
        },
        Sort = new L
ist<SortSpec>() { ne
w SortSpec
        {
                Dire
ction = "ASC",
                Colu
mn = "Name"
        }}
};

searchResults = conn
ector.Roles.FindRole
s(payload);
```

"searchResults" object will contain the same data you can see in the JSON response on the left.

```
        {
            "Id": 38,
            "Name": "Experience Engagement Assigner",
            "Description": "This role will allow assign
ed users the ability to assign Engagements to any Manag
ement Group that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
8.97Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
8.97Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 30,
            "Name": "Experience User",
            "Description": "The Experience User role al
lows users assigned to view access all areas of Experie
nce.",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
7.833Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
7.893Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 44,
            "Name": "Nomad Administrators",
            "Description": null,
            "CreatedTimestampUtc": "2021-10-07T13:42:1
3.523Z",
            "ModifiedTimestampUtc": "2021-10-07T13:42:1
3.523Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 42,
            "Name": "Patch Success Administrator",
            "Description": "The Patch Success Administr
```

Direct Consumer API call	C# code using Consumer SDK library

```
ator role allows users assigned to access all areas of
Patch Success including deploying Patches.",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
9.223Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
9.223Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 26,
            "Name": "Patch Success User",
            "Description": "The Patch Success User role
allows users assigned to view access all areas of Patch
Success and initiate check status and update status.",
            "CreatedTimestampUtc": "2021-10-07T10:51:3
7.49Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:3
7.893Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        }
    ]
}
```

You can also retrieve a singel Role by its Id, here we'll retrieve the Global Administrators role which has the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Roles/1 will yield following response:
 	
Return payload Expand source
 	
`{`
`  "Id": 1,`
`  "Name": "Full Administrator",`
`  "Description": "The Full Administrator has permissions in the system.",`
`  "CreatedTimestampUtc": "2021-10-07T10:5`
	Use Roles object inside the Tachyon connector instance.
	
	Retrieving a specifc Role by its Id Expand source
	
	`roles = connector.Roles.Get(1);`
	
	"role" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
1:34.837Z",
    "ModifiedTimestampUtc": "2021-10-07T10:
51:37.893Z",
    "SystemRole": true,
    "CanBeDelegated": false,
    "NumberOfAssignments": 1,
    "HasSecurityPermission": true
}
```

Lastly, we'll look at retrieving all Roles given Principal has assigned to them using that Principal's Id. Here we'll get all the Roles that "SomeDomain\Jane.Doe" has assigned and that Principal's Id is 3:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Roles/Principal/3 will yield following response: **Return payload** Expand source

```
[
    {
        "PrincipalId": 3,
        "RoleId": 7,
        "CreatedTimestampUtc": "2021-10-08T1
3:39:01.5694234Z",
        "Role": {
            "Id": 7,
            "Name": "All Instructions Action
er",
            "Description": "This role will a
llow assigned users to execute all instructi
ons that are defined as Actions.",
            "CreatedTimestampUtc": "2021-10-
07T10:51:35.063Z",
            "ModifiedTimestampUtc": "2021-10
-07T10:51:37.893Z",
            "SystemRole": true,
            "CanBeDelegated": false,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": false
        },
        "Principal": null
    }
]
```

Use Roles object inside the Tachyon connector instance.

Retrieving all Roles fora given Principa l Expand source

```
roles = connector.Roles.GetForP
rincipal(principalId);
```

"roles" object will contain the same data you can see in the JSON response on the left.

2.5.8.4.3 Getting Securable Types

You can retrieve all Securable Types with following call:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/SecurableTypes will yield following response:
 	
Return payload Expand source
 	
[
{
"Id": 1,
"Name": "InstructionSet",
"CreatedTimestampUtc": "2021-10-07T10:51:34.83Z",
"ModifiedTimestampUtc": "2021-10-07T10:51:36.647Z",
"Operations": [
{
"Id": 2,
"OperationName": "Actioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 4,
"OperationName": "Approver",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 3,
"OperationName": "Questioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 1,
"OperationName": "Viewer",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
}
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Retrieving all Securable Types Expand source
	
	`secTypes = connector.SecurableTypes.GetAll();`
	
	"secTypes" object will contain the same data you can see in the JSON response on the left.

```
        ],
        "IsGlobal": false,
        "Description": "Execute, schedule, cance
l, and approve instructions - view responses"
    },
    {
        "Id": 2,
        "Name": "Security",
        "CreatedTimestampUtc": "2021-10-07T10:51:
34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:34.83Z",
        "Operations": [
            {
                "Id": 7,
                "OperationName": "Delete",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 5,
                "OperationName": "Read",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 6,
                "OperationName": "Write",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            }
        ],
        "IsGlobal": false,
        "Description": "Add and remove Users - vi
ew all Roles - add, modify, and delete Custom rol
es - assign roles to users - view Audit informati
on log"
    },
    {
        "Id": 3,
        "Name": "InstructionSetManagement",
        "CreatedTimestampUtc": "2021-10-07T10:51:
34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:36.647Z",
        "Operations": [
            {
```

```
                "Id": 9,
                "OperationName": "Add",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instruction
SetManagement"
            },
            {
                "Id": 8,
                "OperationName": "Delete",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instruction
SetManagement"
            },
            {
                "Id": 10,
                "OperationName": "Read",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instruction
SetManagement"
            }
        ],
        "IsGlobal": true,
        "Description": "Upload product packs - ad
d, modify, and delete instruction sets - delete i
nstruction definitions"
    },
    {
        "Id": 4,
        "Name": "Infrastructure",
        "CreatedTimestampUtc": "2021-10-07T10:51:
34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:34.83Z",
        "Operations": [
            {
                "Id": 85,
                "OperationName": "Delete",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastruct
ure"
            },
            {
                "Id": 11,
                "OperationName": "Read",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastruct
ure"
```

```
            },
            {
                "Id": 84,
                "OperationName": "Write",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastruct
ure"
            }
        ],
        "IsGlobal": true,
        "Description": "View System health and Sy
stem information - view, add, and edit global set
tings"
    },
    {
        "Id": 5,
        "Name": "CustomProperty",
        "CreatedTimestampUtc": "2021-10-07T10:51:
35.33Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:35.33Z",
        "Operations": [
            {
                "Id": 12,
                "OperationName": "Read",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPrope
rty"
            },
            {
                "Id": 13,
                "OperationName": "Write",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPrope
rty"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and dele
te Custom properties"
    },
    {
        "Id": 6,
        "Name": "Consumer",
        "CreatedTimestampUtc": "2021-10-07T10:51:
35.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
```

```
1:35.877Z",
        "Operations": [
            {
                "Id": 14,
                "OperationName": "Read",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            },
            {
                "Id": 15,
                "OperationName": "Write",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and dele
te Consumers"
    },
    {
        "Id": 7,
        "Name": "AgentDeployment",
        "CreatedTimestampUtc": "2021-10-07T10:51:
36.84Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:36.84Z",
        "Operations": [
            {
                "Id": 18,
                "OperationName": "Approve",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeploy
ment"
            },
            {
                "Id": 16,
                "OperationName": "Execute",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeploy
ment"
            },
            {
                "Id": 17,
                "OperationName": "View",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeploy
ment"
```

```
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, and cancel
1E Client deployment jobs"
    },
    {
        "Id": 8,
        "Name": "AgentInstallerManagement",
        "CreatedTimestampUtc": "2021-10-07T10:51:
36.88Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:36.88Z",
        "Operations": [
            {
                "Id": 19,
                "OperationName": "Add",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstal
lerManagement"
            },
            {
                "Id": 20,
                "OperationName": "Delete",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstal
lerManagement"
            },
            {
                "Id": 21,
                "OperationName": "Read",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstal
lerManagement"
            }
        ],
        "IsGlobal": true,
        "Description": "View, upload, and delete
1E Client installers"
    },
    {
        "Id": 9,
        "Name": "ManagementGroup",
        "CreatedTimestampUtc": "2021-10-07T10:51:
36.98Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:36.98Z",
```

```
        "Operations": [
            {
                "Id": 29,
                "OperationName": "Delete",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementG
roup"
            },
            {
                "Id": 22,
                "OperationName": "Read",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementG
roup"
            },
            {
                "Id": 24,
                "OperationName": "Synchronize",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementG
roup"
            },
            {
                "Id": 23,
                "OperationName": "Write",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementG
roup"
            }
        ],
        "IsGlobal": false,
        "Description": "Create, delete, edit, and
initiate synchronization of Management Groups"
    },
    {
        "Id": 10,
        "Name": "Connector",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 27,
                "OperationName": "Delete",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
```

```
            },
            {
                "Id": 28,
                "OperationName": "Execute",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            },
            {
                "Id": 25,
                "OperationName": "Read",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            },
            {
                "Id": 26,
                "OperationName": "Write",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delet
e, and test Connectors"
    },
    {
        "Id": 11,
        "Name": "Schedule",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 32,
                "OperationName": "Delete",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 30,
                "OperationName": "Read",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 31,
                "OperationName": "Write",
```

```
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and d
elete Schedules - view Schedule history"
    },
    {
        "Id": 12,
        "Name": "ProcessLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 90,
                "OperationName": "Delete",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            },
            {
                "Id": 33,
                "OperationName": "Read",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            },
            {
                "Id": 89,
                "OperationName": "Write",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            }
        ],
        "IsGlobal": true,
        "Description": "View and purge the Proces
s log, Cancel all actions"
    },
    {
        "Id": 13,
        "Name": "SynchronizationLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
```

```
            {
                "Id": 34,
                "OperationName": "Read",
                "SecurableTypeId": 13,
                "SecurableTypeName": "Synchroniza
tionLog"
            }
        ],
        "IsGlobal": true,
        "Description": "View Sync log"
    },
    {
        "Id": 14,
        "Name": "Component",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 35,
                "OperationName": "Read",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component"
            },
            {
                "Id": 86,
                "OperationName": "Write",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component"
            }
        ],
        "IsGlobal": true,
        "Description": "View and configure Compon
ents"
    },
    {
        "Id": 15,
        "Name": "ProviderConfiguration",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 38,
                "OperationName": "Delete",
```

```
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderCon
figuration"
            },
            {
                "Id": 36,
                "OperationName": "Read",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderCon
figuration"
            },
            {
                "Id": 37,
                "OperationName": "Write",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderCon
figuration"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and d
elete Providers"
    },
    {
        "Id": 16,
        "Name": "InfrastructureLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 39,
                "OperationName": "Read",
                "SecurableTypeId": 16,
                "SecurableTypeName": "Infrastruct
ureLog"
            }
        ],
        "IsGlobal": true,
        "Description": "View Infrastructure log"
    },
    {
        "Id": 17,
        "Name": "Repository.Inventory",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.38Z",
```

```
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.38Z",
        "Operations": [
            {
                "Id": 43,
                "OperationName": "Archive",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            },
            {
                "Id": 42,
                "OperationName": "Delete",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            },
            {
                "Id": 53,
                "OperationName": "EvaluateManagem
entGroups",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            },
            {
                "Id": 44,
                "OperationName": "Populate",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            },
            {
                "Id": 40,
                "OperationName": "Read",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            },
            {
                "Id": 41,
                "OperationName": "Write",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.
Inventory"
            }
        ],
```

```
        "IsGlobal": true,
        "Description": "View, create, edit, and d
elete Inventory repositories - populate and archi
ve them"
    },
    {
        "Id": 18,
        "Name": "Application",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.433Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.433Z",
        "Operations": [
            {
                "Id": 46,
                "OperationName": "Delete",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application
"
            },
            {
                "Id": 45,
                "OperationName": "Write",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application
"
            }
        ],
        "IsGlobal": true,
        "Description": "Install and uninstall Tac
hyon Portal applications"
    },
    {
        "Id": 19,
        "Name": "GuaranteedState",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.463Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.463Z",
        "Operations": [
            {
                "Id": 49,
                "OperationName": "Delete",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedS
tate"
            },
```

```
            {
                "Id": 47,
                "OperationName": "Read",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedS
tate"
            },
            {
                "Id": 48,
                "OperationName": "Write",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedS
tate"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and dele
te Rules, Fragments, Trigger templates, and Polic
ies - view Guaranteed State dashboards"
    },
    {
        "Id": 20,
        "Name": "Repository.Patch",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.49Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.49Z",
        "Operations": [
            {
                "Id": 50,
                "OperationName": "Read",
                "SecurableTypeId": 20,
                "SecurableTypeName": "Repository.
Patch"
            }
        ],
        "IsGlobal": true,
        "Description": "View Patch Success dashbo
ards"
    },
    {
        "Id": 21,
        "Name": "Repository.BI",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.5Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.5Z",
```

```
        "Operations": [
            {
                "Id": 52,
                "OperationName": "Populate",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.
BI"
            },
            {
                "Id": 51,
                "OperationName": "Read",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.
BI"
            }
        ],
        "IsGlobal": true,
        "Description": "View and populate the BI
respository"
    },
    {
        "Id": 22,
        "Name": "EventSubscription",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.74Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.74Z",
        "Operations": [
            {
                "Id": 56,
                "OperationName": "Delete",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscr
iption"
            },
            {
                "Id": 54,
                "OperationName": "Read",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscr
iption"
            },
            {
                "Id": 55,
                "OperationName": "Write",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscr
```

```
iption"
            }
        ],
        "IsGlobal": false,
        "Description": "View, create, edit, and d
elete the configurations of event subscriptions"
    },
    {
        "Id": 24,
        "Name": "Experience",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.833Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.833Z",
        "Operations": [
            {
                "Id": 58,
                "OperationName": "Read",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Experience"
            }
        ],
        "IsGlobal": true,
        "Description": "View Experience dashboard
s"
    },
    {
        "Id": 25,
        "Name": "Inventory",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.85Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.85Z",
        "Operations": [
            {
                "Id": 60,
                "OperationName": "Export",
                "SecurableTypeId": 25,
                "SecurableTypeName": "Inventory"
            },
            {
                "Id": 59,
                "OperationName": "Read",
                "SecurableTypeId": 25,
                "SecurableTypeName": "Inventory"
            }
        ],
```

```
        "IsGlobal": true,
        "Description": "View Inventory dashboards
and export inventory data"
    },
    {
        "Id": 26,
        "Name": "Inventory.Association",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.85Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.85Z",
        "Operations": [
            {
                "Id": 64,
                "OperationName": "Delete",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.A
ssociation"
            },
            {
                "Id": 63,
                "OperationName": "Export",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.A
ssociation"
            },
            {
                "Id": 61,
                "OperationName": "Read",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.A
ssociation"
            },
            {
                "Id": 62,
                "OperationName": "Write",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.A
ssociation"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and d
elete SCCM Associations in Inventory"
    },
    {
        "Id": 27,
```

```
        "Name": "AppClarity.Entitlement",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.877Z",
        "Operations": [
            {
                "Id": 68,
                "OperationName": "Delete",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.
Entitlement"
            },
            {
                "Id": 69,
                "OperationName": "Execute",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.
Entitlement"
            },
            {
                "Id": 67,
                "OperationName": "Export",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.
Entitlement"
            },
            {
                "Id": 65,
                "OperationName": "Read",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.
Entitlement"
            },
            {
                "Id": 66,
                "OperationName": "Write",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.
Entitlement"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delet
e, export, and manage AppClarity Entitlement"
    },
    {
```

```
        "Id": 28,
        "Name": "AppClarity.Compliance",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.877Z",
        "Operations": [
            {
                "Id": 73,
                "OperationName": "Delete",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.
Compliance"
            },
            {
                "Id": 74,
                "OperationName": "Execute",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.
Compliance"
            },
            {
                "Id": 72,
                "OperationName": "Export",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.
Compliance"
            },
            {
                "Id": 70,
                "OperationName": "Read",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.
Compliance"
            },
            {
                "Id": 71,
                "OperationName": "Write",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.
Compliance"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delet
e, export, and manage AppClarity Compliance and L
DC"
```

```
    },
    {
        "Id": 29,
        "Name": "AppClarity.Reclaim",
        "CreatedTimestampUtc": "2021-10-07T10:51:
37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:37.877Z",
        "Operations": [
            {
                "Id": 78,
                "OperationName": "Delete",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.
Reclaim"
            },
            {
                "Id": 79,
                "OperationName": "Execute",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.
Reclaim"
            },
            {
                "Id": 77,
                "OperationName": "Export",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.
Reclaim"
            },
            {
                "Id": 75,
                "OperationName": "Read",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.
Reclaim"
            },
            {
                "Id": 76,
                "OperationName": "Write",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.
Reclaim"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delet
```

```
e, export, and manage AppClarity Reclaim"
    },
    {
        "Id": 30,
        "Name": "Engagements",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.01Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.01Z",
        "Operations": [
            {
                "Id": 82,
                "OperationName": "Delete",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements
"
            },
            {
                "Id": 83,
                "OperationName": "Execute",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements
"
            },
            {
                "Id": 80,
                "OperationName": "Read",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements
"
            },
            {
                "Id": 81,
                "OperationName": "Write",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements
"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delet
e, and enable Engagements (Surveys and Announceme
nts)"
    },
    {
        "Id": 31,
        "Name": "VDI",
```

```
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.1Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.1Z",
        "Operations": [
            {
                "Id": 87,
                "OperationName": "Read",
                "SecurableTypeId": 31,
                "SecurableTypeName": "VDI"
            },
            {
                "Id": 88,
                "OperationName": "Write",
                "SecurableTypeId": 31,
                "SecurableTypeName": "VDI"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and d
elete application servers"
    },
    {
        "Id": 32,
        "Name": "ProviderOperationLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.13Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.13Z",
        "Operations": [
            {
                "Id": 91,
                "OperationName": "Read",
                "SecurableTypeId": 32,
                "SecurableTypeName": "ProviderOpe
rationLog"
            }
        ],
        "IsGlobal": true,
        "Description": "Update, delete and view p
rovider configurations"
    },
    {
        "Id": 33,
        "Name": "PolicyDeployment",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.15Z",
```

```
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.15Z",
        "Operations": [
            {
                "Id": 92,
                "OperationName": "Execute",
                "SecurableTypeId": 33,
                "SecurableTypeName": "PolicyDeplo
yment"
            }
        ],
        "IsGlobal": true,
        "Description": "Deploy all types of polic
ies (including metrics, events, and engagements)"
    },
    {
        "Id": 34,
        "Name": "OffloadingData",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.25Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.25Z",
        "Operations": [
            {
                "Id": 93,
                "OperationName": "Offload",
                "SecurableTypeId": 34,
                "SecurableTypeName": "OffloadingD
ata"
            }
        ],
        "IsGlobal": true,
        "Description": "Offload (forward) event d
ata to any Web API responsible for processing tha
t data"
    },
    {
        "Id": 35,
        "Name": "Nomad",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.287Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.287Z",
        "Operations": [
            {
                "Id": 96,
                "OperationName": "Delete",
```

```
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            },
            {
                "Id": 94,
                "OperationName": "Read",
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            },
            {
                "Id": 95,
                "OperationName": "Write",
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            }
        ],
        "IsGlobal": true,
        "Description": "View Nomad dashboards and
SSD peer data. View, add, and delete pre-cache jo
bs. Pause and resume download activity of Nomad c
lients"
    },
    {
        "Id": 36,
        "Name": "Protect",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.317Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.317Z",
        "Operations": [
            {
                "Id": 97,
                "OperationName": "Read",
                "SecurableTypeId": 36,
                "SecurableTypeName": "Protect"
            },
            {
                "Id": 98,
                "OperationName": "Write",
                "SecurableTypeId": 36,
                "SecurableTypeName": "Protect"
            }
        ],
        "IsGlobal": true,
        "Description": "View and deploy patches a
t all endpoints"
    },
```

Direct Consumer API call	C# code using Consumer SDK library


```
    {
        "Id": 37,
        "Name": "PolicyAssignment",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.603Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.603Z",
        "Operations": [
            {
                "Id": 99,
                "OperationName": "Assign",
                "SecurableTypeId": 37,
                "SecurableTypeName": "PolicyAssig
nment"
            }
        ],
        "IsGlobal": false,
        "Description": "Assign Guaranteed State p
olicies to Management Groups"
    },
    {
        "Id": 38,
        "Name": "EngagementAssignment",
        "CreatedTimestampUtc": "2021-10-07T10:51:
38.957Z",
        "ModifiedTimestampUtc": "2021-10-07T10:5
1:38.957Z",
        "Operations": [
            {
                "Id": 100,
                "OperationName": "Assign",
                "SecurableTypeId": 38,
                "SecurableTypeName": "EngagementA
ssignment"
            }
        ],
        "IsGlobal": false,
        "Description": "Assign Engagements (Surve
ys and Announcements) to Management Groups"
    }
]
```


You can also retrieve a specific Securable Type by either its Name or Id.

Direct Consumer API call	C# code using Consumer SDK library

If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/1
 	
If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/Name/InstructionSet
 	
Both yield following response:
 	
Return payload Expand source
 	
`{ "Id": 1, "Name": "InstructionSet", "CreatedTimestampUtc": "2019-11-07T13:14:52.75Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.113Z", "IsGlobal": false, "Description": "Execute, schedule, cancel, and approve instructions - view responses" "Operations": [{ "Id": 2, "OperationName": "Actioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 4, "OperationName": "Approver", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 3, "OperationName": "Questioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 1, "OperationName": "Viewer", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	To retrieve a Securable Type using its Id use:
	
	Retrieving a Securable Type by Id Expand source
	
	`secType = connector.SecurableTypes.Get(1);`
	
	To retrieve a Securable Type using its Name use:
	
	Retrieving Securable Type by Name Expand source
	
	`secType = connector.SecurableTypes.Get("InstructionSet");`
	
	"secType" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    ]
}
```

2.5.8.4.4 Getting Applicable Operations

Applicable Operations work in the context of a Securable Type, which is why Tachyon allows you to retrieve all Applicable Operations for given Securable Type. You can use either Name or Id to specify which Securable Type you want Applicable Operations for.

Direct Consumer API call	C# code using Consumer SDK library

If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableTypeId/1
 	
If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableType Name/InstructionSet
 	
Both yield the following response:
 	
Return payload Expand source
 	
`[{ "Id": 2, "OperationName": "Actioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 4, "OperationName": "Approver", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 3, "OperationName": "Questioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet"`
	Use ApplicableOperations object inside the Tachyon connector instance.
	
	To retrieve Applicable Operations for a Securable Type using the type's Id use:
	
	Retrieve Applicable Operations using Securable Type's Id Expand source
	
	`operations = connector.ApplicableOperations.Get(1);`
	
	To retrieve Applicable Operations for a Securable Type using the type's Name use:
	
	Retrieve Applicable Operations using Securable Type's Name Expand source
	
	`operations = connector.ApplicableOperations.Get("InstructionSet");`
	
	"operations" object will contain the same data you can see in the JSON response on the left.

```
    },
    {
        "Id": 1,
        "OperationName": "Viewer",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionS
et"
    }
]
```

2.5.8.4.5 Getting Permissions

You can retrieve permissions in several different ways, for example for a Principal, a Role or a Securable Type.

Some APIs perform checks for the calling user by pulling the user information from the HTTP request itself. Other APIs allow you to specify which object you want to get permissions for.

2.5.8.4.5.1 Getting permissions for a Principal

Getting all Permissions

Retrieving a principal's permissions is done using the account name (for example "somedomain\jane.doe") of that principal. When directly using the API, you have to encode that account name into base64 before sending it. C# Consumer API SDK will do the encoding for you.

In general, any GET endpoint requires you to base64 encode the principal name, due to the fact that principal names can contain characters that are not allowed in URIs.

When you request permissions for a specific Principal, you'll get the permissions that Principal has stemming from any of their roles.

Please note that these Permissions will specify which Management Group they have been granted on, which reflect on how the Role containing these Permissions is assigned to the Principal.

If given Principal has the same Permission on two different Management Groups (for instance because those Permissions come from two different Roles, which have been assigned using a different Management Group, or perhaps the same Role assigned twice, each time using a different Management Group), there will be two rows for the same Permission which will differ in the Management Group Id and Name.

Another important thing to remember is that a non-global securable type will yield extra Permission row(s), one per child of the Management Group that the original Permission has been assigned on.

What this means is that in an example we saw earlier, if "Actioner" Role is assigned on "Europe" Management Group to a Principal, asking about that Principal's Permission with yield 4 rows - one for "Europe", one for "UK", one for "France" and one for "Italy". This is because "InstructionSet" is a local Securable Type.

Had "Actioner" has Permisison for, for instance, "InfrastructureLog" Securable Type, which is global, and was assigned to "Europe", there would only be one row, the one containing "Europe".

The following examples use the "somedomain\jane.doe" account.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ== will yield following response:
 	
Return payload Expand source
 	
[{
"ManagementGroupId": 1,
"ManagementGroupName": "All Devices",
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 4,
"SecurableTypeName": "Instrumentation",
"RoleId": 4,
"RoleName": "Infrastructure Administrators",
"Allowed": true,
"Operations": [{
"PermissionId": 19,
"OperationId": 11,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-06-07T15:12:40.757Z",
"ModifiedTimestampUtc": "2019-06-07T15:12:40.757Z"
}]
},
{
"ManagementGroupId": 1,
"ManagementGroupName": "All Devices",
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 12,
"SecurableTypeName": "ProcessLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [{
"PermissionId": 61,
"OperationId": 33,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-06-07T15:12:44.13Z",
"ModifiedTimestampUtc": "2019-06-07T15:12:44.13Z"
}]
	Use Permissions object inside the Tachyon connector instance.
	
	Retrieving all Permissions for a specific Principal Expand source
	
	`permissions = connector.Permissions.GetForPrincipal("somedomain\\jane.doe");`
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

```
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 13,
    "SecurableTypeName": "SynchronizationLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 62,
        "OperationId": 34,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 14,
    "SecurableTypeName": "Component",
    "RoleId": 17,
    "RoleName": "Component Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 64,
        "OperationId": 35,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 16,
```

Direct Consumer API call	C# code using Consumer SDK library

```
    "SecurableTypeName": "InfrastructureLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 63,
        "OperationId": 39,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
}]
```

Checking for a specific Permission

You can also retrieve permissions given a Principal has on a particular Securable Type.

In the example below, we'll look at InstructionSet related permissions "somedomain\jane.doe" account has.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ==/Type/InstructionSet will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving InstructionSet related permissions for a Principal** Expand source "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1,
        "RoleName": "Global Administrators",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer",
```

```
permissions = connector.Permissions.GetForPrincipalAndType("somedomain\\jane.doe", "InstructionSet");
```

```
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
    },
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
    }
]
```

If given Securable Type supports Instances, you can also check a Principal's permissions on a specific instance.

In the example below we'll use a different account called "somedomain\john.doe" and check what permissions it has on Instruction Set with the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/Type/InstructionSet/1 will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving Principal's permissions to a specific instruction set** Expand source

```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": 1,
        "SecurableName": "MySet",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-02
T12:04:04.963Z",
                "ModifiedTimestampUtc": "2020-01-0
2T12:04:04.963Z"
```

```
permissions = connector.P
ermissions.GetForPrincipa
lAndTypeAndInstance("some
domain\\john.doe", "Instr
uctionSet", 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
            }
        ]
    }
]
```

If the given Principal has no permissions or does not have permissions on given Securable Type and/or instance, an empty collection is returned.

Checking Permissions on a specific Management Group

You can check what Permissions a Principal has on a given Management Group.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/ManagementGroup/1 will yield following response: **Return payload** Expand source `[{ "ManagementGroupId": 1, "ManagementGroupName": "All Devices", "SecurableId": 1, "SecurableName": "MySet", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTimestampU`	Use Permissions object inside the Tachyon connector instance. **Retrieving Principal's permissions on a specific Management Group** Expand source `permissions = connector.Permissions.GetForPrincipalAndManagementGroup("somedomain\\john.doe", 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
tc": "2020-01-02T12:04:04.963Z"
            }
        ]
    }
]
```

You can also check if given Principal has a specific Permission on a specific Management Group

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/ManagementGroup/1/Type/InstructionSet will yield following response: **Return payload** Expand source [{ "ManagementGroupId": 1, "ManagementGroupName": "All Devices", "SecurableId": 1, "SecurableName": "MySet", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTime	Use Permissions object inside the Tachyon connector instance. **Retrieving a specific type or Permission that a Principal has on a specific Management Group** Expand source `permissions = connector.Permissions.GetForPrincipalAndTypeAndManagementGroupAndInstance("somedomain\\john.doe", 1, 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
stampUtc": "2020-01-02T12:04:
04.963Z"
            }
        ]
    }
]
```

2.5.8.4.5.2 Getting permissions for a Role

You can retieve Role's permissions by using the role Id.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16 will yield following response:

Return payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 12,
        "SecurableTypeName": "ProcessLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 61,
                "OperationId": 33,
                "OperationName": "Read",
                "CreatedTimestampUtc": "201
9-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc": "20
19-11-07T13:14:56.673Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 13,
        "SecurableTypeName": "Synchronizati
onLog",
        "RoleId": 16,
```

Use Permissions object inside the Tachyon connector instance.

Retrieving all permissions assigned to a Role Expand source

```
permissions = connector.Permissi
ons.GetForRole(16);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library


```
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 62,
                "OperationId": 34,
                "OperationName": "Read",
                "CreatedTimestampUtc": "201
9-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc": "20
19-11-07T13:14:56.673Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 16,
        "SecurableTypeName": "Infrastructur
eLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 63,
                "OperationId": 39,
                "OperationName": "Read",
                "CreatedTimestampUtc": "201
9-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc": "20
19-11-07T13:14:56.673Z"
            }
        ]
    }
]
```


You can also retrieve all permissions given Role has on a specific Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16/Type/ProcessLog will yield following response:	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions that a Role has on a given Securable Type** Expand source

Direct Consumer API call	C# code using Consumer SDK library

Return payload Expand source
 	
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 12,
        "SecurableTypeName": "ProcessLog
",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 61,
                "OperationId": 33,
                "OperationName": "Read",
                "CreatedTimestampUtc": "
2019-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc":
"2019-11-07T13:14:56.673Z"
            }
        ]
    }
]
	permissions = connector.Permission
	s.GetForRoleAndType(16, "ProcessLog
	");
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

And for Securable Types that support Instances, you can also check Permissions on a specific Instance.

Here we'll get permissions for a custom Role that's been assigned a 'Viewer' permissions on an Instruction Set.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/30/Type/InstructionSet/1 will yield following response:
 	
Return payload Expand source
 	
[
    {
        "SecurableId": 1,
        "SecurableName": "Wszystko",
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instructi
onSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
	Use Permissions object inside the Tachyon connector instance.
	
	Retrieving permissions a Role has on a specific Instance of a Securable Type Expand source
	
	permissions = connector.Permissions.
	GetForRoleAndTypeAndInstance(30, "In
	structionSet", 1);
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewe
r",
                "CreatedTimestampUtc":
"2020-01-02T12:04:04.963Z",
                "ModifiedTimestampUtc":
"2020-01-02T12:04:04.963Z"
            }
        ]
    }
]
```

2.5.8.4.5.3 Getting permissions for a Securable Type

You can retrieve permissions granted on a Securable Type using the type's Id. This will return a collection of all permissions on a given type assigned to any of the Roles.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Securable/1 will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving all Permisisons on a given Securable Type** Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 1,
        "RoleName": "Global Administrator
s",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
```

```
permissions = connector.Permission
s.GetForSecurableType(1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
```

```
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 6,
        "RoleName": "Global Questioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 21,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 7,
        "RoleName": "Global Actioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 22,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
```

```
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 8,
        "RoleName": "Global Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 23,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 27,
        "RoleName": "ServiceNow ITSM Conn
ect",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 136,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:18:05.767Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:18:05.767Z"
```

Direct Consumer API call	C# code using Consumer SDK library

```
            }
        ]
    }
]
```

You can also retrieve permissions granted on a specific Instance of a Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Securable/1/1 will yield following response: **Return payload** Expand source `[{ "SecurableId": 1, "SecurableName": null, "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTimestampUtc": "2020-01-02T12:04:04.963Z" }] }]`	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions assigned on a specific instance of a Securable Type** Expand source `connector.Permissions.GetForSecurableTypeAndInstance(1, 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

2.5.8.4.5.4 Helper APIs

Consumer API has a number of endpoints that are not directly involved with its RBAC system but instead provide various utility functions around the general principal and permission area.

Who am I?

You can ask the API to return the information about the user who's making the request. The API examines the incoming request, retrieves the name of the user from it, makes sure the user is authenticated, and returns information summary about the user, like user's principal name, SID, display name and email.

This endpoint is useful for systems that cannot assume they're running under a particular account, like a browser or an application running in the context of launching user. In those cases this endpoint can be used to obtain the name of the user, which is then fed to the Permissions endpoint seen above in order to retrieve the permissions of the caller.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/whoami will yield following response: **Return payload** Expand source `{ "PrincipalName": "Somedomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-19970", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "Photo": null }`	Use PrincipalSearch object inside the Tachyon connector instance. **Who am I?** Expand source `user = connector.PrincipalSearch.GetCurrentlyLoggedInUser();` "user" object will contain the same data you can see in the JSON response on the left.

Active Directory search

Because in order to add a Principal to Tachyon you have to know the SID of the user, Tachyon's Consumer API exposes endpoints that help you look up users in Active Directory and retrieve information about them, including their SID.

You can perform a search for Active Directory accounts whose common name or sAMAccountName contain given phrase. Details of what exact search is performed are beyond the scope of this page and depend on type of objects being searched for, like users, groups and computers and whether your Tachyon installation is using GC or LDAP.

In the examples below we'll be looking for accounts that contain the phrase "Administrator". How you perform this search depends on your version of Tachyon.

Tachyon versions up to and including 4.1

You simply provide a base64 encoded search string and issue a GET request.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/QWRtaW5pc3RyYXRvcg== will yield following response: **Return payload** Expand source [{ "PrincipalName": "SomeDomain\\CSUserAdministrator", "ExternalId": "S-1-5-21-1202660629-789336058-1353024091-8145", "Email": null, "DisplayName": "CSUserAdministrator", "IsGroup": true }, { "PrincipalName": "SomeDomain\\CSVoiceAdministrator", "ExternalId": "S-1-5-21-1202660629-789336058-1343023091-8024", "Email": null, "DisplayName": "CSVoiceAdministrator", "IsGroup": true }, { "PrincipalName": "SomeDomain\\Exchange Organization Administrators", "ExternalId": "S-1-5-21-1202660629-789336058-1343324091-6881", "Email": null, "DisplayName": "Exchange Organization Administrators", "IsGroup": true }, { "PrincipalName": "SomeDomain\\Exchange Public Folder Administrators", "ExternalId": "S-1-5-21-1202660629-789336058-1340024091-7747", "Email": null, "DisplayName": "Exchange Public Folder Administrators", "IsGroup": true	Use PrincipalSearch object inside the Tachyon connector instance. **Searching for principals in Active Directory** Expand source `results = connector.PrincipalSearch.SearchForPrincipals("Administrator");` "results" object will contain the same data you can see in the JSON response on the left.

```
        },
        {
                "PrincipalName": "SomeDomain\\Exc
hange Recipient Administrators",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange Recipien
t Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exc
hange View-Only Administrators",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange View-Onl
y Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Sec
urity Administrator",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security Administ
rator",
                "IsGroup": true
        }
]
```


Here you have greater control over the search. A POST request should be used and in its payload you can specify, apart from the search string, number of results returned, sort column and order.

You can search for "user" and "group" object types and sort in either ascending (using "ASC") or descending order (using "DESC") on following columns: "cn", "mail", "sAMAccountName", "description", "objectSid", "displayName". Column names, object types and sorting direction string are not case sensitive.

You have to provide the SearchText (the text to search for) and at least one object type. Pagesize and sort are optional and if not provided will default to respectively 100 and ascending sorting on display name.

One things that you have to keep in mind when it comes to PageSize property is that, due to internal implementation, it defines the size of the page returned by Active Directory. This means that you can get fewer results than you requested, even if there would have been more to return because Tachyon filters out users already in the system. As an example, if you define the page size to be 10 and the Active Directory has 18 entries

that match your search string, it will return 10 entries. Now let us assume that out of those 10 entries 1 user is already in Tachyon. This user will be filtered out of the return data set, so you will receive only 9 entries.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/PrincipalSearch with following payload:
 	
Request payload Expand source
 	
`{ "SearchText": "Administrator", "ObjectTypes": ["user", "group"], "PageSize": 100, "Sort": { "Column": "displayname", "Direction": "ASC" } }`
 	
will yield following response:
 	
Return payload Expand source
 	
`[{ "PrincipalName": "SomeDomain\\CSUserAdministrator", "ExternalId": "S-1-5-21-1202660629-789336058-1353024091-8145", "Email": null, "DisplayName": "CSUserAdministrator", "IsGroup": true }, { "PrincipalName": "SomeDomain\\CSVoiceAdministrator", "ExternalId": "S-1-5-21-1202660629-789336058-1343023091-8024", "Email": null, "DisplayName": "CSVoiceAdministrator", "IsGroup": true }, { "PrincipalName": "SomeDomain\\Exchange Organization Administrato`
	Use PrincipalSearch object inside the Tachyon connector instance.
	
	Retrieving 100 entries sorted by displayname Expand source
	
	`var searchParams = new ActiveDirectorySearchModel { SearchText = "Administrator", ObjectTypes = new List<string> { "user", "group" }, PageSize = 100, Sort = new SortSpec { Column = "displayname", Direction = "ASC" } };`
	
	`results = connector.PrincipalSearch.SearchForPrincipals(searchParams);`
	
	"results" object will contain the same data you can see in the JSON response on the left.

```
rs",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343324091-6881",
                "Email": null,
                "DisplayName": "Exchange
Organization Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Public Folder Administrat
ors",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-7747",
                "Email": null,
                "DisplayName": "Exchange
Public Folder Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Recipient Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange
Recipient Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange View-Only Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange
View-Only Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Security Administrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security
```

Direct Consumer API call	C# code using Consumer SDK library

```
Administrator",
                "IsGroup": true
        }
]
```

2.5.8.4.5.5 Retrieving Active Directory information about a specific account

You can also retrieve user's information by their principal name, though this endpoint only returns authenticated users who are also Principals in Tachyon (either directly or throught group membership).

This means that in order to obtain information about somedomain\jane.doe using this endpoint, somedomain\jane.doe has to be a domain account that's either a Principal in Tachyon, or belongs to an Active Directory group that is a Principal in Tachyon.

When using the API directly, the principal name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/user/c29tZWRvbWFpblxqYW5lLmRvZQ==will yield following response: **Return payload** Expand source `[ { "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "jane.doe@somedomain.com", "DisplayName": "Jane Doe", "IsGroup": false } ]`	Use PrincipalSearch object inside the Tachyon connector instance. **Retrieve information about a principal from Active Directory** Expand source `results = connector.PrincipalSearch.SearchForUser("SomeDomain\\Jane.Doe");` "results" object will contain the same data you can see in the JSON response on the left.

Retrieving members of an Active Directory group

You can ask Tachyon to retrieve members of an Active Directory group. This group does not have to be a principal in Tachyon for this functionality to work.

When using the API directly the group name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

In the example below we'll use an AD group called "Tachyon users" and retrieve its members.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/GetMembers/VGFjaHlvbiB1c2Vycw==will yield following response:
 	
Return payload Expand source
 	
[
{
"PrincipalName": "SomeDomain\\Jane.Doe",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842",
"Email": "jane.doe@somedomain.com",
"DisplayName": "Jane Doe",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Meetra.Surnik",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-8631",
"Email": "Meetra.Surnik@somedomain.com",
"DisplayName": "Meetra Surnik",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Keiran.Halcyon",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-1138",
"Email": "Keiran.Halcyon@somedomain.com",
"DisplayName": "Keiran Halcyon",
"IsGroup": false
}
]
	Use PrincipalSearch object inside the Tachyon connector instance.
	
	Retrieving members of an Aciv e Directory group Expand source
	
	`results = connector.PrincipalSearch.GetGroupMembers("Tachyon users");`
	
	"results" object will contain the same data you can see in the JSON response on the left.

2.5.8.5 Configuring Tachyon RBAC through the Consumer API

In this section we will look at configuring RBAC emphasizing operations on Principals, Roles and Permissions but also covering Securable Types and Operations later on.

2.5.8.5.1 Adding Principals from Active Directory

Adding Principals to Tachyon will be one of the first things you'll do after installing the system. A fresh installation of Tachyon (which in this context means an installation where there was no previous Master database or such database was dropped) will have two Principals - the account

used to install the Tachyon Server (and created the database to be specific) and NT AUTHORITY\Network Service account.

These accounts have limited permissions so to properly use the system you should add more Principals. In order to add a principal to Tachyon you will need an External Id, also called SID, for an active directory account that you wish to add as a principal. Although you can obtain account details through various means. In Active Directory search we've seen Tachyon exposes endpoints that return this information and enables you to search for Active Directory accounts.

Here we'll assume you already have the account details and focus on adding an account as a Principal to Tachyon.

In the example below we'll "SomeDomain\Jane.Doe" account and add it to Tachyon.

To add an account as a Principal you'll at the very least need to supply an ExternalId (SID) and Principal Name, though its advised to also supply a Display Name. Also, unless you set the Enabled flag to true, the newly created account will be disabled by default.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Principals with following payload:
 	
Request payload Expand source
 	
`{ "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "IsGroup": false, "Enabled": true }`
 	
Will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": false,`
	Use Principals object inside the Tachyon connector instance.
	
	Adding a Principal to Tachyon Expand source
	
	`var payload = new Principal { PrincipalName = "SomeDomain\\Jane.Doe", ExternalId = "S-1-5-21-1202660629-789336058-1343024091-23842", Email = "Jane.Doe@SomeDomain.com", DisplayName = "Jane Doe", IsGroup = false, Enabled = true };`
	
	`principal = connector.Principals.Add(payload);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```

Note

You cannot add a Principal with the same PrincipalName or ExternalId as another Principal that's already in the system, because both of those values must be unique.

Updating a Principal

Updating a Principal is very similar to adding one. You have to use a PUT verb instead of a POST if using the API directly, or Update method instead of Add if you're using the SDK. You also have to provide an Id that belongs to an existing Principal. Remember, you cannot modify system Principals.

Again, you have to remember that ExternalId and PrincipalName must remain unique so you cannot change Principal's PrincipalName or ExternalId to one that matches another Principal's.

2.5.8.5.2 Configuring Roles

Roles are the pivotal point of an RBAC system.

Tachyon comes with a number of pre-defined roles, which should be sufficient to start with, but in time you will most likely create custom roles, either to restrict access to instruction sets or to cover sets of permission required by specific roles within your organization.

Adding, Editing and Removing a Role

First let's look at creating a Role. This can be done either in isolation, where just the role is created and any permissions must be assigned in subsequent call(s), or as a more complete package, where a Role is created along with a set of Permissions, which can include Management Groups.

Regardless of which approach is used, Principals have to be assigned separately.

To create a Role all you require is a name, and this name must be unique. Description can optionally be provided and you can't create system roles.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roleswith following payload: **Request payload** Expand source `{ "Name": "Custom role", "Description": "this is a description" }`	Use Roles object inside the Tachyon connector instance. **Creating a Role** Expand source `var payload = new Role { Name = "Custom role", Description = "this is a description" };`

Direct Consumer API call	C# code using Consumer SDK library
Will yield following response: **Return payload** Expand source `{ "Id": 31, "Name": "Custom role", "Description": "this is a description", "CreatedTimestampUtc": "2020-01-21T13:36:55.8956916Z", "ModifiedTimestampUtc": "2020-01-21T13:36:55.8956916Z", "SystemRole": false }`	`role = connector.Roles.Add(payload);` "role" object will contain the same data you can see in the JSON response on the left.

Creating a Role with Permissions is possible by joining Role creation we've just seen with creating permissions, which is described a bit later.

This effectively means that we take a regular Role creation payload and add a 'Permissions' property to it. That propery is an array of permissions, which we are yet to look at in greater details.

The example below creates a role, which has "Viewer" and "Approver" permissions on Instruction Set with the Id of 1 and "Actioner", "Questioner", "Viewer" and "Approver" permissions on Instruction Set with the Id of 2. It also has permissions on three Management Groups.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Completewith following payload: **Request payload** Expand source `{ "Name": "Complete Role 1", "Description": "This is a test role", "Permissions": [{ "SecurableId": 2, "SecurableTypeId": 1, "Allowed": true, "Operations": [{ "OperationId": 2 }, { "OperationId": 4 },`	Use Roles object inside the Tachyon connector instance. **Creating a Role with Permissions** Expand source `var payload = new Tachyon.SDK.Consumer.Models.Send.RoleWithPermissionsAndManagementGroups { Name = "Complete Role 1", Description = "This is a test role", Permissions = new AggregatedPermission[] { new AggregatedPermi`

Direct Consumer API call

C# code using Consumer SDK library

```
        {
            "OperationId": 3
        },
        {
            "OperationId": 1
        }]
    },
    {
        "SecurableId": 1,
        "SecurableTypeId": 1,
        "Allowed": true,
        "Operations": [{
            "OperationId": 1
        },
        {
            "OperationId": 4
        }]
    }]
}
```

Will yield following response:

Return payload Expand source

```
{
    "Permissions": [
        {
            "SecurableId": 1,
            "SecurableName": null,
            "SecurableTypeId": 1,
            "SecurableTypeName": "InstructionSet
",
            "RoleId": 32,
            "RoleName": "Complete Role 1",
            "Allowed": true,
            "Operations": [
                {
                    "PermissionId": 142,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 143,
```

```
ssion
        {
            SecurableId =
2,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 2 },
                new Permiss
ionOperation { OperationId
= 3 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        },
        new AggregatedPermi
ssion
        {
            SecurableId =
1,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        }
    }
};
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"OperationId": 4,
"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
}
]
},
{
"SecurableId": 2,
"SecurableName": null,
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet",
"RoleId": 32,
"RoleName": "Complete Role 1",
"Allowed": true,
"Operations": [
{
"PermissionId": 138,
"OperationId": 2,
"OperationName": "Actioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 139,
"OperationId": 4,
"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 140,
"OperationId": 3,
"OperationName": "Questioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},</pre>
	<pre>role = connector.Roles.Add(payload);</pre>
	"role" object will contain the same data you can see in the JSON response on the left.

```
                {
                    "PermissionId": 141,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                }
            ]
        }
    ]
    "Id": 32,
    "Name": "Complete Role 1",
    "Description": "This is a test role",
    "CreatedTimestampUtc": "2020-01-22T07:18:57.
7082854Z",
    "ModifiedTimestampUtc": "2020-01-22T07:18:5
7.7082854Z",
    "SystemRole": false
}
```


To update a Role you should use PUT versions of the two endpoints seen above and provide one additional property in the JSON payload - the Id of the Role being modified. If you're using the Consumer SDK, you should call Update method instead of Add and you must provide an Id in the payload object sent to the API.

Since the names of Roles have to be unique, you won't be able to change the name of a Role to one that is the same as another Role.

Changing the Role details with https://my.tachyon.server/Consumer/Roles (or using the Update method in Consumer SDK that takes Role object type) will change just the details of the role will not change the permissions in any way. It will not change which Mangement Groups or Principals are assigned to the Role either.

Changing the Role via https://my.tachyon.server/Consumer/Roles/Complete will not change which Principals are assigned to the Role, but it will change the Permissions and Management Groups. When using this endpoint with PUT verb (or using the Update method in Consumer SDK that takes RoleWithPermissionsAndManagementGroups object type), all of the Permissions for given Role will be replaced with Permissions provided in the payload and all Management Groups will be replaced with Management Groups provided in the payload. This means that if you do not provide Permissions (omit the property, provide a null value or an empty array), all Permissions will be removed from the Role. Likewise, if you do not provide ManagementGroups all Management Groups will be unassigned from the Role.

Deleting a role is straightforward and you just need its Id. You can also delete multiple Roles in one call by providing a collection of Role Ids. In either case, you cannot delete a system Role.

Direct Consumer API call	C# code using Consumer SDK library

To delete a single Role make a DELETE request to https://my.tachyon.server/Consumer/Roles/31 ,which will delete the Role with the Id of 31.
 	
To delete multiple Roles, make a DELETE request and send an array of Ids to https://my.tachyon.server/Consumer/Roles. To delete Roles with Ids 31, 32, 33 and 34 you would send this payload:
 	
Request payload Expand source
 	
`[31,32,33,34]`
	Use Roles object inside the Tachyon connector instance.
	
	To delete a single Role:
	
	Deleting the Role with Id 31 Expand source
	
	`connector.Roles.Delete(31);`
	
	To delete multiple Roles:code
	
	Deleting Roles with Ids 31,32,33 and 34 . Expand source
	
	`connector.Roles.DeleteMultiple(new List<int> {31, 32, 33, 34});`

Adding and removing Permissions to a Role

The main method of assigning Permissions is to send two collections, one with Permissions that need to be changed or updated and another one with permissions that should be removed.

Crafting the correct payload should be done carefully as omitting an entry can result in deletion of an existing Permission.

PermissionsToSaveOrUpdate collection should have permissions you either want to add or modify. You only need to provide Ids, names are ignored. It is important to understand when the system considers given operation an 'updated' and when a 'creation' as it will behave slightly differently.

Because a Permission is, effectively, a link between a Securable Type and a Role, with possible addition of Securable Id, those three properties define it. Operations are considered a property of a Permission and they are always considered together - i.e. a single Permission can Allow or Deny a set of Operations, but it cannot mix and match.

When you request a Permission to be added or updated, Consumer API will look at the SecurableId, SecurableTypeId and RoleId properties provided and check for any permissions already existing for this combination of values. If none are found, a new permission is added with the operations specified in the payload. If an entry is found, then Operations are examined next. If the existing Permission has Operations that aren't present in the Permission sent to the API, they are removed. Any operations present in the Permission sent to the API that do not exist in the Permission that is already in the system are added.

As you might have noticed, PermissionsToSaveOrUpdate can be used to remove Permissions by omitting Operations you want to remove. Furthermore, if you provide a Permission without any Operations, the entire Permission is removed.

Lets look at an example that should help us better understand how this functionality works.

Let us assume that we have a Permission on Instruction Set "My set" assigned to Role "Custom Role" with just one Operation "View". "My Set" has the Id of 4, "Custom Role" has the Id of 31, "Instruction Set" Securable Type has the Id of 1 and "Viewer" has the Id of 1. This means that the existing Permission will look as follows:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        }
        ]
}
```

Now let us assume that a request comes into the Consumer API that looks as follows:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 1
                        },
                                {
                                "OperationId": 3
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

This request matches a Permission we already have, because it has the same SecurableId, SecurableTypeId and RoleId, but the Operations are slightly different - Id of 3 has been added, which represents 'Questioner' Operation. Following the rules outlined above, the API will match this to the existing Permission and see that Operation with the Id of 1 already exists, so no action is needed but the Operation with the Id of 3 doesn't exist, so it needs to be added. Afterwards we will end up with a permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
```

```
    "Operations": [
                {
                "OperationId": 1
        },
                {
                "OperationId": 3
        }
        ]
}
```

If now we were to make another request with with following payload:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 2
                        },
                                {
                                "OperationId": 4
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

The Consumer API will again recognise that a Permission already exists that matches the one in the payload and will examine the Operations. Because neither operation 1 nor 3, which do exist in the Permission already in the system are present in the payload, they will be removed. Operations 2 and 4 (Actioner and Approve respectively) do not exist in the Permission so they will be added and changed made will result in a Permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 2
        },
                {
```

```
                "OperationId": 4
        }
        ]
}
```

As a last example let's look at what will happen when we send this payload to the API:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": []
        }
    ],
    "PermissionsToDelete": []
}
```

Again, the system will recognise this as an existing Permission and will remove Operations 2 and 4 because they do not exist in the request payload. But since there are no Operations to add, the entire Permission will be removed.

Lastly, let's look at how an example API call would look like:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Permissions with following payload:	Use Permissions object inside the Tachyon connector instance.
Payload sent to https://my.tachyon.server/Consumer/Permissions Expand source	**Creating Permissions** Expand source
`{ "PermissionsToSaveOrUpdate": [ { "Allowed": true, "SecurableTypeId": 5, "SecurableId": null, "RoleId": "31", "Operations": [ { "OperationId": 12 }, { "OperationId": 13`	`var payload = new Permi ssionsSaveContainer { PermissionsToSaveOr Update = new List<Aggre gatedPermission> { new AggregatedP ermission { RoleId = 3 1, SecurableTy`

Direct Consumer API call	C# code using Consumer SDK library

```
                }
            ]
        }
    ],
    "PermissionsToDelete": []
}
```

will yield following response:

Response payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 5,
        "SecurableTypeName": "CustomProperty",
        "RoleId": 31,
        "RoleName": "Custom Role",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 191,
                "OperationId": 12,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
14:47:13.023Z"
            },
            {
                "PermissionId": 192,
                "OperationId": 13,
                "OperationName": "Write",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
14:47:13.023Z"
            }
        ]
    }
]
```

```
peId = 5,
            Operations
= new List<PermissionOp
eration>
            {
                new Per
missionOperation
                {
                    Ope
rationId = 12
                },
                new Per
missionOperation
                {
                    Ope
rationId = 13
                }
            }
        }
    }
};
permissions = connecto
r.Permissions.AddOrUpda
te(payload);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

2.5.8.5.3 Configuring Assignments

An assignment is a link between a Principal, a Role and a Management Group and we have already briefly explored how they work.

Before we look at how you can use the API to manage assignments, let us go over basic rules and limitations:

- A single assignment is between one Principal, one Role and one Management Group. If you, for instance, wish to assign the same principal to the same role but for multiple Management Groups, you will have to create multiple assignments
- You cannot change assignments of a System Principal.
- If you are a global security administrator, you can use any Role in an assignments. If you are a local security administrator, you can only use delegatable Roles in your assignments.
- You can manipulate any assignments which uses a Management Group which you have "Security" → "Write" Permission on. You will be able to see other assignments but they are marked as "Inaccessible" (see their "AccessType" property).
- You can only assign "Full Administrator" Role to "All Devices" Management Group.
- You cannot assign "Group Administrator" Role to "All Devices" Management Group.
- You cannot assign any Role that has a Permission which uses "Security" securable type to a Management Group which you have "ReadOnly" access to (see Management Group's "AccessType" property"), unless that Management Group is "All Devices".

Now that's out of the way, let us look at two ways in which we can create assignments.

The first one is to use the bulk add functionality by issuing a POST request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups API.

This API takes a collection of assignment objects and will attemp to create them. Any assignments that already exist are ommited. Any assignments that are invalid will cause the entire operation to fail and return an error.

Here's an example of how you'd create assignments using that API:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups with following payload: **Payload sent to** https://my.tachyon.server/Consumer/Roles/ManagementGroups Expand source `[ { "PrincipalId": 3, "RoleId": 40, "ManagementGroupId": 2 }, { "PrincipalId": 4, "RoleId": 37, "ManagementGroupId": 10`	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance. **Assigning Management Groups to a Role** Expand source `var payload = new List<PrincipalRoleManagementGroup> { new PrincipalRoleManagementGroup { PrincipalId = 3, RoleId = 4`

Direct Consumer API call

C# code using Consumer SDK library

```
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 11
    }
]
```

which will yield a response similar to this one:

Return payload Expand source

```
[
    {
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 2,
        "CreatedTimestampUtc": "2021-10-19T15:10:
17.927Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-72
8399001-2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Do
e",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T1
5:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T1
5:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 1
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Administrat
or role allows users assigned to assign users per
mission that have been defined as delegatable in
Tachyon, this means that they can be restricted t
o a Management Group.",
            "CreatedTimestampUtc": "2021-10-18T1
1:40:29.79Z",
            "ModifiedTimestampUtc": "2021-10-18T1
```

```
0,
                ManagementG
roupId = 2
        },
        new PrincipalRoleMa
nagementGroup
        {
                PrincipalId
= 4,
                RoleId = 3
7,
                ManagementG
roupId = 10
        },
        new PrincipalRoleMa
nagementGroup
        {
                PrincipalId
= 4,
                RoleId = 3
7,
                ManagementG
roupId = 11
        }
};

var assignments = connecto
r.PrincipalRoleManagementGr
oups.AddAssignments(payloa
d);
```

"assignments" object will contain the same data you can see in the JSON response on the left.

```
1:40:29.79Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 2,
            "Name": "UK",
            "Description": "All computers in UK",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "UK",
            "HashOfMembers": "9670A13F-C5C6-4819-
AE77-4D7D8983180C",
            "CreatedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
            "ParentUsableId": "global",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 10,
        "CreatedTimestampUtc": "2021-10-19T15:10:
17.927Z",
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326578-72
8399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\John.Do
e",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
```

```
            "NumberOfAssignments": 2
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assi
gner",
            "Description": "This role will allow
assigned users the ability to assign Policies to
any Management Group that is including in their a
ssignments.",
            "CreatedTimestampUtc": "2021-10-18T1
1:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T1
1:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 10,
            "Name": "USWestServers",
            "Description": "All servers in West U
S",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "ServersWestUS",
            "HashOfMembers": "DD46B223-02DE-4839-
BFC6-1CA3F9E2EFBA",
            "CreatedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 11,
        "CreatedTimestampUtc": "2021-10-19T15:10:
17.927Z",
        "Principal": {
```

```
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326578-72
8399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\John.Do
e",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
            "NumberOfAssignments": 2
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assi
gner",
            "Description": "This role will allow
assigned users the ability to assign Policies to
any Management Group that is including in their a
ssignments.",
            "CreatedTimestampUtc": "2021-10-18T1
1:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T1
1:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
            "Name": "USWestDesktops",
            "Description": "All desktops in West
US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-
BEC0-1CC22AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
```

```
            "ModifiedTimestampUtc": "2021-10-19T1
5:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```

While this is useful to create a lot of assignments in bulk, but in real life operations you will usually deal with Assignments for a single instance of a particular type of entity, so either a Principal, a Role or a Management Group.

There are separate APIs that allow you to replace all Assignments for either a Principal, Role or a Management Group. Each of those APIs will check if you're not violating rules outlined above and that all assignments are valid.

What is important is to remember that you should provide the API with the Assignments that are supposed to exist AFTER the call has finished, not merely the ones you want to add. These API endpoints will make a diff between what you provided and what exists in the DB and decide what to add and what to remove.

Because of this, if you load Assignments for a Role and you do not have Permission to some of those Assignments, you can still add more Assignments for that Role or remove those Assignments you do have Permission on, as long as list of assignment you send to the API contains all assignments that should exist, that should be kept, so including the ones you do not have Permission to.

That way the API will recognise those entries as already existing and will not attempt to delete them.

As an example, imagine that Role with Id 3 has following assignments:

- Principal Id 1, Management Group 1
- Principal Id 3, Management Group 2
- Principal Id 7, Management Group 1

and you want to have Principal Id 3 have assignment to this Role on Management Group 4, not 2. You have Permission to manipulate assignment for Management Groups 2, 3 and also 4, but do not have Permission to manipulate assignments for Management Group 1.

To do that you'd send following assignments to appropriate endpoint:

- Principal Id 1, Management Group 1
- Principal Id 3, Management Group 4
- Principal Id 7, Management Group 1

Because you have not provided the assignment for Principal Id 3 and Management Group 2, this Assignment will be deleted, which is ok because you have Permission to do it.

And becuase you provided an Assignment for Principapl Id 3 and Management Group 4 and it doesn't exist, it will be created because you have Permission to do it.

Now, if you were to provide only this assignment to the API:

- Principal Id 3, Management Group 4

the system would determine that Assignments for Principal Id 1 / Mamagement Group 1 and Principal Id 7 / Management Group 1 are missing from the list you provided so you wish to delete them. but since you do not have Permission to perform such an operation, you would get an error.

Let us now look at the replacement API endpoints.

First, we have an endpoints that allows us to replace all Assignments for a specific Principal:

Direct Consumer API call	C# code using Consumer SDK library

You can identify the Principal either by Id by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Principal/Id/3
 	
or by Name by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Principal/Name/U29tZURvbWFpblxKYW5lLkRvZQ== . The name must be base64 encoded.
 	
In either case, the payload is the same:
 	
Request payload Expand source
 	
`[ { "RoleId": 40, "ManagementGroupId": 2 }, { "RoleId": 37, "ManagementGroupId": 10 }, { "RoleId": 37, "ManagementGroupId": 11 } ]`
 	
Either call will yield following result:
 	
Return payload Expand source
 	
`[ {`
	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.
	
	Replacing Assignments for a Principal Expand source
	
	`IEnumerable<PrincipalRoleManagementGroup> assignments; var payload = new List<PrincipalRoleManagementGroup> { new PrincipalRoleManagementGroup { RoleId = 40, ManagementGroupId = 2 }, new PrincipalRoleManagementGroup { R`

Direct Consumer API call	C# code using Consumer SDK library

<pre>"PrincipalId": 3,
"RoleId": 37,
"ManagementGroupId": 10,
"CreatedTimestampUtc": "2021-10-20T15:41:54.93Z",
"Principal": {
"Id": 3,
"ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1010",
"PrincipalName": "SomeDomain\\Jane.Doe",
"Email": null,
"Enabled": true,
"CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
"ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
"SystemPrincipal": false,
"DisplayName": "Stranger",
"IsGroup": false,
"NumberOfAssignments": 3
},
"Role": {
"Id": 37,
"Name": "Guaranteed State Policy Assigner",
"Description": "This role will allow assigned users the ability to assign Policies to any Management Group that is including in their assignments.",
"CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
"ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
"SystemRole": true,
"CanBeDelegated": true,
"NumberOfAssignments": 4,
"HasSecurityPermission": false,
"AccessType": null
},
"ManagementGroup": {
"Id": 10,
"Name": "USWestServers",
"Description": "All servers in West US",
"Expression": null,
"TachyonManagementGroupType": 1,
"TachyonDeviceCount": 10,
"UsableId": "ServersWestUS",
"HashOfMembers": "DD46B223-02DE-4839-BFC6-1CA3F9E2EFBA",
"CreatedTimestampUtc": "2021-10-19T15:07:51.11</pre>
	<pre>oleId = 37,
	M
	anagementGroupId = 10
	},
	new Princ
	ipalRoleManagemen
	tGroup
	{
	R
	oleId = 37,
	M
	anagementGroupId = 11
	}
	};
	
	// To replace ass
	ignments for a Pr
	incipal using Pri
	ncipal's Id:
	assignments = con
	nector.PrincipalR
	oleManagementGrou
	ps.ReplaceAssignm
	entsForPrincipal
	(3, payload);
	
	// to replace ass
	ignments for a Pr
	incipal using Pri
	ncipal's Name:
	assignments = con
	nector.PrincipalR
	oleManagementGrou
	ps.ReplaceAssignm
	entsForPrincipal
	("SomeDomain\\Jan
	e.Doe", payload);</pre>
	"assignments" object will contain the same data you can see in the JSON response on the left.

```
Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.1
1Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 11,
        "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-2
836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.67
7Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.6
77Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned
users the ability to assign Policies to any Management Gro
up that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.50
7Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.5
07Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
```

```
            "Name": "USWestDesktops",
            "Description": "All desktops in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-BEC0-1CC2
2AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11
Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.1
1Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 2,
        "CreatedTimestampUtc": "2021-10-19T15:10:17.927Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-2
836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.67
7Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.6
77Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Administrator role a
llows users assigned to assign users permission that have
been defined as delegatable in Tachyon, this means that th
ey can be restricted to a Management Group.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.79
Z",
```

Direct Consumer API call	C# code using Consumer SDK library

```
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.7
9Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": true,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 2,
            "Name": "UK",
            "Description": "All computers in UK",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "UK",
            "HashOfMembers": "9670A13F-C5C6-4819-AE77-4D7D
8983180C",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11
Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.1
1Z",
            "ParentUsableId": "global",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```

Next, we have endpoints that allow you to replace all Assignments for a specific Role

Direct Consumer API call	C# code using Consumer SDK library
You can identify the Role either by Id by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Role/Id/37 or by Name by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Role/Name/R3VhcmFudGVlZCBTdGF0ZSBQb2xpY3kgQXNzaWduZXI= . The name must be base64 encoded. In either case, the payload is the same:	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance. **Replacing Assignments for a Role** Expand source

Direct Consumer API call

C# code using Consumer SDK library

Request payload Expand source

```
[
        {
                "PrincipalId": 3,
                "ManagementGroupId": 10
        },
        {
                "PrincipalId": 3,
                "ManagementGroupId": 11
        }
]
```

Either call will yield following result:

Return payload Expand source

```
[
    {
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 10,
        "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z
",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001
-2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.
677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:2
1.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigne
d users the ability to assign Policies to any Management
Group that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.
```

```
IEnumerable<Principa
lRoleManagementGrou
p> assignments;
var payload = new Li
st<PrincipalRoleMa
nagementGroup>
{
        new Princip
alRoleManagementGro
up
        {
                Pri
ncipalId = 3,
                Man
agementGroupId = 11
        },
        new Princip
alRoleManagementGro
up
        {
                Pri
ncipalId = 3,
                Man
agementGroupId = 10
        }
};

// To replace assig
nments for a Role u
sing Role's ID:
assignments = conne
ctor.PrincipalRoleM
anagementGroups.Rep
laceAssignmentsForR
ole(37, payload);

// To replace assig
nments for a Role u
sing Role's Name;
assignments = conne
ctor.PrincipalRoleM
anagementGroups.Rep
laceAssignmentsForR
ole("Guaranteed Sta
te Policy Assigner
", payload);
```

Direct Consumer API call	C# code using Consumer SDK library

507Z",
"ModifiedTimestampUtc": "2021-10-18T11:40:2
9.507Z",
"SystemRole": true,
"CanBeDelegated": true,
"NumberOfAssignments": 4,
"HasSecurityPermission": false,
"AccessType": null
},
"ManagementGroup": {
"Id": 10,
"Name": "USWestServers",
"Description": "All servers in West US",
"Expression": null,
"TachyonManagementGroupType": 1,
"TachyonDeviceCount": 10,
"UsableId": "ServersWestUS",
"HashOfMembers": "DD46B223-02DE-4839-BFC6-1C
A3F9E2EFBA",
"CreatedTimestampUtc": "2021-10-19T15:07:51.
11Z",
"ModifiedTimestampUtc": "2021-10-19T15:07:5
1.11Z",
"ParentUsableId": "WestUS",
"AccessType": "Unknown"
},
"AccessType": "ReadWrite"
},
{
"PrincipalId": 3,
"RoleId": 37,
"ManagementGroupId": 11,
"CreatedTimestampUtc": "2021-10-20T15:41:54.93Z
",
"Principal": {
"Id": 3,
"ExternalId": "S-1-5-21-3276326578-728399001
-2836074973-1010",
"PrincipalName": "SomeDomain\\Jane.Doe",
"Email": null,
"Enabled": true,
"CreatedTimestampUtc": "2016-11-30T15:03:21.
677Z",
"ModifiedTimestampUtc": "2016-11-30T15:03:2
1.677Z",
"SystemPrincipal": false,
"DisplayName": "Stranger",
	"assignments" object will contain the same data you can see in the JSON response on the left.

```
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigne
d users the ability to assign Policies to any Management
Group that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.
507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:2
9.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
            "Name": "USWestDesktops",
            "Description": "All desktops in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-BEC0-1C
C22AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.
11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:5
1.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```


Finally, we have endpoints that allow you to replace all Assignments for a specific Management Group.

Direct Consumer API call

You can identify the Role either by Id by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/ManagementGroup/Id/9

or by Name by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/ManagementGroup/UsableId/USEastDesktops

In either case, the payload is the same:

Request payload Expand source

```
[
        {
                "PrincipalId": 3,
                "RoleId": 40
        },
        {
                "PrincipalId": 4,
                "RoleId": 37
        },
        {
                "PrincipalId": 3,
                "RoleId": 37
        }
]
```

Either call will yield the same result:

Return payload Expand source

```
[
    {
        "IsInherited": false,
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T
12:29:51.523Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326
578-728399001-2836074973-1010",
            "PrincipalName": "SomeDomain\\J
ane.Doe",
```

C# code using Consumer SDK library

Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.

Replacing all Assignments for a Management Group Expand source

```
IEnumerable<PrincipalRoleManagem
entGroup> assignments;
var payload = new List<Principal
RoleManagementGroup>
{
        new PrincipalRoleManagem
entGroup
        {
                PrincipalId = 3,
                RoleId = 40
        },
        new PrincipalRoleManagem
entGroup
        {
                PrincipalId = 4,
                RoleId = 37
        },
        new PrincipalRoleManagem
entGroup
        {
                PrincipalId = 3,
                RoleId = 37
        }
};

// To replace assignments for a
Management Group using Managemen
t Group Id:
assignments = connector.Principa
lRoleManagementGroups.ReplaceAss
ignmentsForManagementGroup(9, pa
yload).ReceivedObject;

// To replace assignments for a
Management Groups using Manageme
nt Group's UsableId:
assignments = connector.Principa
lRoleManagementGroups.ReplaceAss
ignmentsForManagementGroup("USEa
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"Email": null,
"Enabled": true,
"CreatedTimestampUtc": "2016-11
-30T15:03:21.677Z",
"ModifiedTimestampUtc": "2016-1
1-30T15:03:21.677Z",
"SystemPrincipal": false,
"DisplayName": "Stranger",
"IsGroup": false,
"NumberOfAssignments": 5
},
"Role": {
"Id": 37,
"Name": "Guaranteed State Polic
y Assigner",
"Description": "This role will
allow assigned users the ability to assign
Policies to any Management Group that is in
cluding in their assignments.",
"CreatedTimestampUtc": "2021-10
-18T11:40:29.507Z",
"ModifiedTimestampUtc": "2021-1
0-18T11:40:29.507Z",
"SystemRole": true,
"CanBeDelegated": true,
"NumberOfAssignments": 4,
"HasSecurityPermission": false,
"AccessType": "ReadWrite"
},
"ManagementGroup": {
"Id": 9,
"Name": "USEastDesktops",
"Description": "All desktops in
East US",
"Expression": null,
"TachyonManagementGroupType":
1,
"TachyonDeviceCount": 10,
"UsableId": "DesktopsEastUS",
"HashOfMembers": "89CABF45-7C93
-4DA6-8C9A-8234557B44E4",
"CreatedTimestampUtc": "2021-10
-19T15:07:51.11Z",
"ModifiedTimestampUtc": "2021-1
0-19T15:07:51.11Z",
"ParentUsableId": "EastUS",
"AccessType": "Unknown"
},</pre>
	<pre>stDesktops", payload).ReceivedOb
	ject;</pre>
	"assignments" object will contain the same data you can see in the JSON response on the left.

```
        "AccessType": "ReadWrite"
    },
    {
        "IsInherited": false,
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T
12:29:51.53Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326
578-728399001-2836074973-1010",
            "PrincipalName": "SomeDomain\\J
ane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11
-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-1
1-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 5
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Admin
istrator role allows users assigned to assi
gn users permission that have been defined
as delegatable in Tachyon, this means that
they can be restricted to a Management Grou
p.",
            "CreatedTimestampUtc": "2021-10
-18T11:40:29.79Z",
            "ModifiedTimestampUtc": "2021-1
0-18T11:40:29.79Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": true,
            "AccessType": "ReadWrite"
        },
        "ManagementGroup": {
            "Id": 9,
            "Name": "USEastDesktops",
```

```
            "Description": "All desktops in
East US",
            "Expression": null,
            "TachyonManagementGroupType":
1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsEastUS",
            "HashOfMembers": "89CABF45-7C93
-4DA6-8C9A-8234557B44E4",
            "CreatedTimestampUtc": "2021-10
-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-1
0-19T15:07:51.11Z",
            "ParentUsableId": "EastUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "IsInherited": false,
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T
12:29:51.53Z",
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326
578-728399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\J
ohn.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11
-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-1
1-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
            "NumberOfAssignments": 1
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Polic
y Assigner",
            "Description": "This role will
allow assigned users the ability to assign
```

```
Policies to any Management Group that is in
cluding in their assignments.",
            "CreatedTimestampUtc": "2021-10
-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-1
0-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": "ReadWrite"
        },
        "ManagementGroup": {
            "Id": 9,
            "Name": "USEastDesktops",
            "Description": "All desktops in
East US",
            "Expression": null,
            "TachyonManagementGroupType":
1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsEastUS",
            "HashOfMembers": "89CABF45-7C93
-4DA6-8C9A-8234557B44E4",
            "CreatedTimestampUtc": "2021-10
-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-1
0-19T15:07:51.11Z",
            "ParentUsableId": "EastUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```

Please note that the replacement APIs take the Id, Name or UsableId used to identify the object from the URI, not from the payload.

This means that if you were to, for instance, replace Assignments for a Principapal and in the payload provide principal Ids, they will be ignored. Likewise for Role Id when replacing Assignments for a Role and Management Group Id when replacing Assignment for a Management Group.

Deleting Assignments

Assignments can also be deleted, providing you have the Permission to do so, either in bulk:

Direct Consumer API call	C# code using Consumer SDK library

Make a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups with following payload:
 	
Request payload Expand source
 	
[
{
"PrincipalId": 3,
"RoleId": 40,
"ManagementGroupId": 2
},
{
"PrincipalId": 4,
"RoleId": 37,
"ManagementGroupId": 10
},
{
"PrincipalId": 4,
"RoleId": 37,
"ManagementGroupId": 11
}
]
 	
No data is returned by this call.
	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.
	
	Deleting Assignments in bulk Expand source
	
	var payload = new List<PrincipalRoleManagementGroup>
	{
	new PrincipalRoleManagementGroup
	{
	PrincipalId = 3,
	RoleId = 40,
	ManagementGroupId = 2
	},
	new PrincipalRoleManagementGroup
	{
	PrincipalId = 4,
	RoleId = 37,
	ManagementGroupId = 10
	},
	new PrincipalRoleManagementGroup
	{
	PrincipalId = 4,
	RoleId = 37,
	ManagementGroupId = 11
	}
	};
	
	connector.PrincipalRoleM

Direct Consumer API call	C# code using Consumer SDK library
	`anagementGroups.DeleteAssignments(payload);`

or individually:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/PrincipalId/3/RoleId/40/ManagementGroupId/2 without any payload. No data is returned by this call.	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance. **Deleting a single assignment** Expand source `connector.PrincipalRoleManagementGroups.DeleteAssignment(3, 40, 2);`

2.5.8.5.4 Configuring Securable Types and Applicable Operations

We have already seen how to retrieve Securable Types and now we'll look at how we can create and modify them.

All you need to create a Securable Type is a name, which must be unique.

In order to delete a Securable Type, you have to first delete all Permissions that use that Securable Types and all Applicable Operations linked to the Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/SecurableTypes	Use SecurableTypes object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

Paylaod sent to https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Name": "Custom securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Custom securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "ModifiedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "Operations": null }`
	Creating a new Securable Type Expand source
	
	`var payload = new SecurableType { Name = "Custom securable type" }; return connector.SecurableTypes.Add(payload);`
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

If you wish to update a securable type you'll have to provide its Id:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/SecurableTypes**Paylaod sent to** https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Id": 26, "Name": "Modified securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Modified securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "ModifiedTimestampUtc": "2020-01-24T14:38:24.8756464Z",`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Modifying an existing Securable Type Expand source
	
	`var payload = new SecurableType { Id = 26, Name = "Modified securable type" }; return connector.S`

Direct Consumer API call	C# code using Consumer SDK library
`"Operations": null` `}`	`ecurableTypes.Upda te(payload);` "securableType" object will contain the same data you can see in the JSON response on the left.

and to delete this Sedurable Type:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/SecurableTypes/26 This call will not yield any response.	Use SecurableTypes object inside the Tachyon connector instance. **Deleting a Securable Type** Expand source `connector.SecurableTypes.D elete(26);`

It is worth noting that Securable Types are created "empty" and that Applicable Operation have to be created separately.

Dealing with Operations

Applicable Operations exist only in the context of a Securable Type. While Applicable Operations must be unique within a single Securable Type, they do not have to be unique between Securable Types.

To create an Operation you have to provide either Id or Name (but not both) of Securable Type for which the Operation should be created. In the example below we'll use the Securable Type we've just created, which has the Id of 26.

Direct Consumer API call	C# code using Consumer SDK library
Make a POST request to https://my.tachyon.server/Consumer/ApplicableOperations to use Securable Type Id	Use ApplicableOperations object inside the Tachyon connector instance. To use Securable Type Id:

Direct Consumer API call	C# code using Consumer SDK library

Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{`
`  "OperationName": "View",`
`  "SecurableTypeId": 26`
`}`
 	
or with this payload to use Securable Type Name:
 	
Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{`
`  "OperationName": "View",`
`  "SecurableTypeName": "Modified securable type"`
`}`
 	
Both will yield following response:
 	
Return payload Expand source
 	
`{`
`  "Id": 63,`
`  "OperationName": "View",`
`  "SecurableTypeId": 26,`
`  "SecurableTypeName": "Modified securable type"`
`}`
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation`
	`{`
	`  SecurableTypeId = 26,`
	`  OperationName = "View"`
	`};`
	`operation = connector.ApplicableOperations.Add(payload);`
	
	To use Securable Type Name:
	
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation`
	`{`
	`  SecurableTypeName = "Modified securable type",`
	`  OperationName = "View"`
	`};`
	`operation = connector.ApplicableOperations.Add(payload);`
	
	"operation" object will contain the same data you can see in the JSON response on the left.

To delete an Operation you just need its Id.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/ApplicableOperations/63	Use ApplicableOperations object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
	Deleting an Applicable Operation Expand source `connector.ApplicableOperations.Delete(63);`

2.5.9 Set up Principals, Roles and Permissions, up to and including Tachyon version 5.2

This section contains an overview of Tachyon role based access control (RBAC) and its components and how to configure Tachyon RBAC. This is not intended as an in-depth explanation of what RBAC is, but as a demonstration of how it's been implemented in Tachyon and how users can configure it programmatically.

The C# examples assume you're using Tachyon Consumer SDK and you have an instantiated an instance of Tachyon connector class in an object called connector. All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API.

In the following examples this detailis leftout, stating that the returned object contains the data.For example, when we say that XYZ object contains certain data, this means the ReceivedObject property inside said object contains that data.

2.5.9.1 Basics of role based access control (RBAC)

Role-based access control is an access control mechanism defined around roles and privileges. This security model pivots around the concept or a Role. Users (called principals in Tachyon) can be assigned to a Role and it's through a Role they gain permissions to perform actions. Each element of Tachyon's security system leads back to a Role.

2.5.9.2 RBAC objects in Tachyon

Tachyon has several types of objects that are part of its RBAC system. This section serves as a brief description of these objects and their purpose.

2.5.9.2.1 Principals

Principals are identical to users. They're not called users because the word 'user' is traditionally associated with a person. A principal is, from a technical standpoint, an Active Directory account which may be a user account or a computer account. It can also be an AD group.

Tachyon allows access only to principals authenticated through Windows authentication and known to Tachyon itself. Depending on the request, Tachyon establishes the permissions of the calling principal by looking at the roles that principal is assigned to. By default, a fresh installation of Tachyon has a principal representing the user account installing it and another principal for the "NT AUTHORITY\Network Service", which many of Tachyon's services will be running as.

2.5.9.2.2 Roles

All permissions lead to roles. Principals must have roles assigned to them before using the system.

2.5.9.2.3 Permissions

A permission is an ability to perform an operation on a securable type.

2.5.9.2.4 Securable Types

A Securable Type represents a type of an object that can have permissions assigned to it. For example, Instructions can have permissions, as can Consumers and Management Groups. Security itself is an object principals need permissions to, so they can modify it, it will also have a Securable Type. If an element of Tachyon has security defined for it, it must have a securable type.

Instances

An Instance is one, specifc copy of an object of a given Securable Type. This can, for instance, be a single Instruction Set.

While most permissions work on Securable Types as a general thing, where if a Principal is assigned a Permission on a given Securable Type they can work with any object of that type, some Securable Types allow Permissions to be specified on just one given instance of a Securable Type object, giving a more granular control over access to that Securable Type.

As of version 5.0 only InstructionSet securable type supports instances.

2.5.9.2.5 Operations

An Operation (also called Applicable Operation) represents the type of an action that can be performed on a securable type, like "Read" or "Write".

2.5.9.2.6 Example how to read permissions

Marc (**Principal**) through his Global Questioners (**Role**) has Questioner (**Applicable Operation**) permission on Instructions (**Securable type**).

A Principal will have a specific permission on given securable type through a role the principal belongs to. This is the only way principals can obtain permissions and permissions are always for a specific operation on a given type.

To establish the full permission set of a given principal you have to combine all permissions from all the roles assigned to that principal.

2.5.9.3 Retrieving RBAC objects

In this section we will look at endpoints that allow you to examine existing RBAC objects.

2.5.9.3.1 Getting Principals

The most basic functionality is to retrieve all Principals:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals will yield following response: **Return payload** Expand source [{ "Id": 1, "ExternalId": "S-1-5-21-1202660629-789336158-1349024091-27850", "PrincipalName": "SomeDomain\\Administrator", "Email": "Administrator@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": true, "DisplayName": "Administrator", "IsGroup": false }, { "Id": 2, "ExternalId": "S-1-5-20", "PrincipalName": "NT AUTHORITY\\Network Service", "Email": null, "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:56.687Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.687Z", "SystemPrincipal": true, "DisplayName": "Network Service", "IsGroup": false }, { "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true,	Use Principals object inside the Tachyon connector instance. **Retrieving all principals** Expand source `principals = onnector.Principals.GetAll();` "principals" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library


```
        "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
        "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
        "SystemPrincipal": false,
        "DisplayName": "Jane Doe",
        "IsGroup": false
    },
    {
        "Id": 4,
        "ExternalId": "S-1-5-21-3276326578-728
399001-2836074973-1009",
        "PrincipalName": "SomeDomain\\John.Doe
",
        "Email": "John.Doe@SomeDomain",
        "Enabled": true,
        "CreatedTimestampUtc": "2016-11-30T15:
10:10.73Z",
        "ModifiedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
]
```


Or just a single Principal by their Id. Here we'll look for "SomeDomain\\Jane.Doe", who in has the Id of 3:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals/3 will yield following response: **Return payload** Expand source	Use Principals object inside the Tachyon connector instance. **Retrieving specific Principal** Expand source `principal = connector.Principals.Get(3);` "principal" object will contain the same data you can see in the JSON response on the left.


```
{
    "Id": 3,
    "ExternalId": "S-1-5-21-1202660629-7893360
58-1343024091-23842",
    "PrincipalName": "SomeDomain\\Jane.Doe",
    "Email": "Jane.Doe@SomeDomain.com",
    "Enabled": true,
    "CreatedTimestampUtc": "2019-11-07T13:14:5
2.777Z",
    "ModifiedTimestampUtc": "2019-11-07T13:14:
52.777Z",
```

Direct Consumer API call	C# code using Consumer SDK library

```
    "SystemPrincipal": false,
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```

You can also retrieve all Principals that have a specific Role assigned to them. Here we'll get all Principals who are "Global Administrators", which has the Id of 1:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Principals/Role/1 will yield following response:
 	
Return payload Expand source
 	
[
{
"PrincipalId": 1,
"RoleId": 1,
"CreatedTimestampUtc": "2019-11-07T13:15:01.533Z",
"Role": null,
"Principal": {
"Id": 1,
"ExternalId": "S-1-5-21-1202660629-789336158-1349024091-27850",
"PrincipalName": "SomeDomain\\Administrator",
"Email": "Administrator@SomeDomain.com",
"Enabled": true,
"CreatedTimestampUtc": "2019-11-07T13:14:52.777Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z",
"SystemPrincipal": true,
"DisplayName": "Administrator",
"IsGroup": false
}
}
]
	Use Principals object inside the Tachyon connector instance.
	
	Retrieving all principals that have a specific role Expand source
	
	principals = connector.Principals.GetForRole(1);
	
	"principals" object will contain the same data you can see in the JSON response on the left.

2.5.9.3.2 Getting Roles

Much like with Principals, you can retrieve all Roles in the system:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles will yield following response: **Return payload** Expand source [{ "Id": 11, "Name": "1E Client Deployment Administrators", "Description": "This role can create, view and cancel 1E Client deployment jobs", "CreatedTimestampUtc": "2019-11-07T13:14:56.28Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.28Z", "SystemRole": true }, { "Id": 12, "Name": "1E Client Installer Administrators", "Description": "This role can upload, delete and view 1E Client installers", "CreatedTimestampUtc": "2019-11-07T13:14:56.307Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.307Z", "SystemRole": true }, { "Id": 23, "Name": "Applications Administrators", "Description": "This role can upload and delete Applications", "CreatedTimestampUtc": "2019-11-07T13:14:56.693Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.693Z", "SystemRole": true }, { "Id": 17, "Name": "Component Viewers", "Description": "This role can view components", "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",	Use Roles object inside the Tachyon connector instance. **Retrieving all Roles** Expand source `roles = connector.Roles.GetAll();` "roles" object will contain the same data you can see in the JSON response on the left.

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 14,
        "Name": "Connector Administrators",
        "Description": "This role can create, update, d
elete and view connectors; and test them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 10,
        "Name": "Consumer Administrators",
        "Description": "This role can manage Consumers
that use the Tachyon platform",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.523
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.52
3Z",
        "SystemRole": true
    },
    {
        "Id": 9,
        "Name": "Custom Properties Administrators",
        "Description": "This role can add, edit or dele
te custom properties",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.133
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.13
3Z",
        "SystemRole": true
    },
    {
        "Id": 28,
        "Name": "Event Subscription Administrators",
        "Description": "This role has full control over
the Event Subscription configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.90
7Z",
        "SystemRole": true
```

```
    },
    {
        "Id": 29,
        "Name": "Event Subscription Viewers",
        "Description": "This role can view the Event Su
bscription configuration and reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.90
7Z",
        "SystemRole": true
    },
    {
        "Id": 7,
        "Name": "Global Actioners",
        "Description": "This role can ask questions, vi
ew responses and send actions for all instruction sets
",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 1,
        "Name": "Global Administrators",
        "Description": "Has the combined rights of all
the other system roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:52.75
7Z",
        "SystemRole": true
    },
    {
        "Id": 5,
        "Name": "Global Approvers",
        "Description": "This role can approve actions f
or all instruction sets, for anyone other than self",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
```

```
        "Id": 6,
        "Name": "Global Questioners",
        "Description": "This role can ask questions and
view responses for all instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 8,
        "Name": "Global Viewers",
        "Description": "This role can view instructions
and responses for all instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 24,
        "Name": "Guaranteed State Administrators",
        "Description": "This role has full control over
the Guaranteed State configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.71
7Z",
        "SystemRole": true
    },
    {
        "Id": 25,
        "Name": "Guaranteed State Viewers",
        "Description": "This role can view the Guarante
ed State configuration and reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.71
7Z",
        "SystemRole": true
    },
    {
        "Id": 4,
        "Name": "Infrastructure Administrators",
        "Description": "This role can view system statu
```

```
s instrumentation and system information",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.49
3Z",
        "SystemRole": true
    },
    {
        "Id": 2,
        "Name": "Instruction Set Administrators",
        "Description": "This role can add product pack
s; add, modify and delete instruction sets; and delete
instruction definitions",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
        "Id": 19,
        "Name": "Inventory Administrators",
        "Description": "This role can create, update, d
elete and view inventory repositories; populate and arc
hive them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 20,
        "Name": "Inventory Viewers",
        "Description": "This role can view inventory re
positories",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 16,
        "Name": "Log Viewers",
        "Description": "This role can view process, syn
chronization and infrastructure logs",
```

```
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 13,
        "Name": "Management Group Administrators",
        "Description": "This role can create, delete an
d update management groups; and initiate synchronizatio
n of management groups",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.4Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.4Z
",
        "SystemRole": true
    },
    {
        "Id": 21,
        "Name": "Management Group Sync Initiators",
        "Description": "This role can view management g
roups and initiate synchronization of management groups
",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 30,
        "Name": "MySet Viewers",
        "Description": null,
        "CreatedTimestampUtc": "2020-01-02T12:02:12.01Z
",
        "ModifiedTimestampUtc": "2020-01-02T12:02:12.01
Z",
        "SystemRole": false
    },
    {
        "Id": 26,
        "Name": "Patch Success Viewers",
        "Description": "This role can view the Patch Su
ccess dashboards",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.73Z
",
```

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.73
Z",
        "SystemRole": true
    },
    {
        "Id": 3,
        "Name": "Permissions Administrators",
        "Description": "This role can add or remove use
rs; view all roles; add, modify  and delete custom role
s; assign roles to any instruction sets and define thei
r permissions; and view the admin log",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
        "Id": 22,
        "Name": "Permissions Readers",
        "Description": "This role can view all roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 18,
        "Name": "Provider Configuration Administrators
",
        "Description": "This role can update, delete an
d view provider configurations",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 15,
        "Name": "Schedule Administrators",
        "Description": "This role can create, update, d
elete and view schedules; and can view schedule history
",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
```

Direct Consumer API call	C# code using Consumer SDK library

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 27,
        "Name": "ServiceNow ITSM Connect",
        "Description": "This role is used with the Serv
iceNow 1E ITSM Connect application. Once the 1E ITSM Co
nnect application is installed in ServiceNow this role
will be used to grant permissions to the specified Serv
ice Account user.",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.743
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.74
3Z",
        "SystemRole": false
    }
]
```

But unlike the Principals, roles also have an endpoint that supports filtering, sorting and paging. In the example below we'll look for custom roles and sort them by name.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Search with following payload: **Request payload** Expand source `{ "PageSize": 10, "Start": 1, "Filter": { "Attribute": "SystemRole", "Operator": "==", "Value": "false" }, "Sort": [{ "Direction": "ASC", "Column": "Name" }] }`	Use Roles object inside the Tachyon connector instance. **Searching for Roles** Expand source `var payload = new Search { PageSize = 10, Start = 1, Filter = new ExpressionObject { Attribute = "SystemRole", Oper`

Direct Consumer API call	C# code using Consumer SDK library

Yields following response:
 	
Return payload Expand source
 	
{
"TotalCount": 2,
"Items": [
{
"SystemRole": false,
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 1,
"Id": 30,
"Name": "MySet Viewers",
"Description": null,
"CreatedTimestampUtc": "2020-01-02T12:02:12.01Z",
"ModifiedTimestampUtc": "2020-01-02T12:02:12.01Z"
},
{
"SystemRole": false,
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 0,
"Id": 27,
"Name": "ServiceNow ITSM Connect",
"Description": "This role is used with the ServiceNow 1E ITSM Connect application. Once the 1E ITSM Connect application is installed in ServiceNow this role will be used to grant permissions to the specified Service Account user.",
"CreatedTimestampUtc": "2019-11-07T13:14:56.743Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.743Z"
}
]
}
	ator = "==",
	Value = "false"
	},
	Sort = new List<SortSpec>() { new SortSpec
	{
	Direction = "ASC",
	Column = "Name"
	}}
	};
	
	searchResults = connector.Roles.FindRoles(payload);
	
	"searchResults" object will contain the same data you can see in the JSON response on the left.

You can also retrieve a singel Role by its Id, here we'll retrieve the Global Administrators role which has the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles/1 will yield following response: **Return payload** Expand source `{ "Id": 1, "Name": "Global Administrators", "Description": "Has the combined rights of all the other system roles", "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.757Z", "SystemRole": true }`	Use Roles object inside the Tachyon connector instance. **Retrieving a specifc Role by its Id** Expand source `roles = connector.Roles.Get(1);` "role" object will contain the same data you can see in the JSON response on the left.

Lastly, we'll look at retrieving all Roles given Principal has assigned to them using that Principal's Id. Here we'll get all the Roles that "SomeDomain\Jane.Doe" has assigned and that Principal's Id is 3:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles/Principal/3 will yield following response: **Return payload** Expand source `[{ "PrincipalId": 3, "RoleId": 2, "CreatedTimestampUtc": "2019-11-07T13:14:55.433Z", "Role": { "SystemRole": true, "AssignedManagementGroupCount": 0, "HasAllDevicesManagementGroupAssigned": false, "AssignedPrincipalCount": 1, "Id": 2, "Name": "Instruction Set Administrators", "Description": "This role can add product packs; add, modify and delete instruction sets; and delete instruction definit`	Use Roles object inside the Tachyon connector instance. **Retrieving all Roles fora given Principal** Expand source `roles = connector.Roles.GetForPrincipal(principalId);` "roles" object will contain the same data you can see in the JSON response on the left.

```
ions",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 3,
        "CreatedTimestampUtc": "2019-11-07T1
3:14:54.963Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAss
igned": false,
            "AssignedPrincipalCount": 1,
            "Id": 3,
            "Name": "Permissions Administrat
ors",
            "Description": "This role can ad
d or remove users; view all roles; add, modi
fy  and delete custom roles; assign roles to
any instruction sets and define their permis
sions; and view the admin log",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 5,
        "CreatedTimestampUtc": "2019-11-07T1
3:15:01.533Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAss
igned": false,
            "AssignedPrincipalCount": 1,
            "Id": 5,
```

Direct Consumer API call	C# code using Consumer SDK library


```
            "Name": "Global Approvers",
            "Description": "This role can ap
prove actions for all instruction sets, for
anyone other than self",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    }
]
```


2.5.9.3.3 Getting Securable Types

You can retrieve all Securable Types with following call:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/SecurableTypes will yield following response: **Return payload** Expand source	Use SecurableTypes object inside the Tachyon connector instance. **Retrieving all Securable Types** Expand source


```
[
    {
        "Id": 1,
        "Name": "InstructionSet",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.113Z",
        "Operations": [
            {
                "Id": 2,
                "OperationName": "Actioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            },
            {
                "Id": 4,
                "OperationName": "Approver",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
```


```
secTypes = connector.Securab
leTypes.GetAll();
```

"secTypes" object will contain the same data you can see in the JSON response on the left.

```
onSet"
            },
            {
                "Id": 3,
                "OperationName": "Questioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            },
            {
                "Id": 1,
                "OperationName": "Viewer",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            }
        ]
    },
    {
        "Id": 2,
        "Name": "Security",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:52.75Z",
        "Operations": [
            {
                "Id": 7,
                "OperationName": "Delete",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 5,
                "OperationName": "Read",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 6,
                "OperationName": "Write",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            }
        ]
    },
    {
```

```
        "Id": 3,
        "Name": "InstructionSetManagement",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.113Z",
        "Operations": [
            {
                "Id": 9,
                "OperationName": "Add",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            },
            {
                "Id": 8,
                "OperationName": "Delete",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            },
            {
                "Id": 10,
                "OperationName": "Read",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            }
        ]
    },
    {
        "Id": 4,
        "Name": "Instrumentation",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:52.75Z",
        "Operations": [
            {
                "Id": 11,
                "OperationName": "Read",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Instrumen
tation"
            }
        ]
    },
```

```
    {
        "Id": 5,
        "Name": "CustomProperty",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:55.13Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:55.13Z",
        "Operations": [
            {
                "Id": 12,
                "OperationName": "Read",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPro
perty"
            },
            {
                "Id": 13,
                "OperationName": "Write",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPro
perty"
            }
        ]
    },
    {
        "Id": 6,
        "Name": "Consumer",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:55.523Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:55.523Z",
        "Operations": [
            {
                "Id": 14,
                "OperationName": "Read",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            },
            {
                "Id": 15,
                "OperationName": "Write",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            }
        ]
    },
    {
```

```
        "Id": 7,
        "Name": "AgentDeployment",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.277Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.277Z",
        "Operations": [
            {
                "Id": 18,
                "OperationName": "Approve",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            },
            {
                "Id": 16,
                "OperationName": "Execute",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            },
            {
                "Id": 17,
                "OperationName": "View",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            }
        ]
    },
    {
        "Id": 8,
        "Name": "AgentInstallerManagement",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.307Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.307Z",
        "Operations": [
            {
                "Id": 19,
                "OperationName": "Add",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            },
            {
                "Id": 20,
```

```
                "OperationName": "Delete",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            },
            {
                "Id": 21,
                "OperationName": "Read",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            }
        ]
    },
    {
        "Id": 9,
        "Name": "ManagementGroup",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.397Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.397Z",
        "Operations": [
            {
                "Id": 29,
                "OperationName": "Delete",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 22,
                "OperationName": "Read",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 24,
                "OperationName": "Synchronize",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 23,
                "OperationName": "Write",
                "SecurableTypeId": 9,
```

```
                "SecurableTypeName": "Managemen
tGroup"
            }
        ]
    },
    {
        "Id": 10,
        "Name": "Connector",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 27,
                "OperationName": "Delete",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 28,
                "OperationName": "Execute",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 25,
                "OperationName": "Read",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 26,
                "OperationName": "Write",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            }
        ]
    },
    {
        "Id": 11,
        "Name": "Schedule",
        "CreatedTimestampUtc": "2019-11-07T13:1
```

```
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 32,
                "OperationName": "Delete",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 30,
                "OperationName": "Read",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 31,
                "OperationName": "Write",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            }
        ]
    },
    {
        "Id": 12,
        "Name": "ProcessLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 33,
                "OperationName": "Read",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLo
g"
            }
        ]
    },
    {
        "Id": 13,
        "Name": "SynchronizationLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
```

```
14:56.663Z",
        "Operations": [
            {
                "Id": 34,
                "OperationName": "Read",
                "SecurableTypeId": 13,
                "SecurableTypeName": "Synchroni
zationLog"
            }
        ]
    },
    {
        "Id": 14,
        "Name": "Component",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 35,
                "OperationName": "Read",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component
"
            }
        ]
    },
    {
        "Id": 15,
        "Name": "ProviderConfiguration",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 38,
                "OperationName": "Delete",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderC
onfiguration"
            },
            {
                "Id": 36,
                "OperationName": "Read",
                "SecurableTypeId": 15,
```

```
                "SecurableTypeName": "ProviderC
onfiguration"
            },
            {
                "Id": 37,
                "OperationName": "Write",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderC
onfiguration"
            }
        ]
    },
    {
        "Id": 16,
        "Name": "InfrastructureLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 39,
                "OperationName": "Read",
                "SecurableTypeId": 16,
                "SecurableTypeName": "Infrastru
ctureLog"
            }
        ]
    },
    {
        "Id": 17,
        "Name": "Repository.Inventory",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 43,
                "OperationName": "Archive",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 42,
                "OperationName": "Delete",
```

```
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 53,
                "OperationName": "EvaluateManag
ementGroups",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 44,
                "OperationName": "Populate",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 40,
                "OperationName": "Read",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 41,
                "OperationName": "Write",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            }
        ]
    },
    {
        "Id": 18,
        "Name": "Application",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.693Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.693Z",
        "Operations": [
            {
                "Id": 46,
                "OperationName": "Delete",
                "SecurableTypeId": 18,
```

```
                "SecurableTypeName": "Applicati
on"
            },
            {
                "Id": 45,
                "OperationName": "Write",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Applicati
on"
            }
        ]
    },
    {
        "Id": 19,
        "Name": "GuaranteedState",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.713Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.713Z",
        "Operations": [
            {
                "Id": 49,
                "OperationName": "Delete",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            },
            {
                "Id": 47,
                "OperationName": "Read",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            },
            {
                "Id": 48,
                "OperationName": "Write",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            }
        ]
    },
    {
        "Id": 20,
        "Name": "Repository.Patch",
        "CreatedTimestampUtc": "2019-11-07T13:1
```

```
4:56.73Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.73Z",
        "Operations": [
            {
                "Id": 50,
                "OperationName": "Read",
                "SecurableTypeId": 20,
                "SecurableTypeName": "Repositor
y.Patch"
            }
        ]
    },
    {
        "Id": 21,
        "Name": "Repository.BI",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.737Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.737Z",
        "Operations": [
            {
                "Id": 52,
                "OperationName": "Populate",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repositor
y.BI"
            },
            {
                "Id": 51,
                "OperationName": "Read",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repositor
y.BI"
            }
        ]
    },
    {
        "Id": 22,
        "Name": "EventSubscription",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.907Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.907Z",
        "Operations": [
            {
                "Id": 56,
```

```
                "OperationName": "Delete",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            },
            {
                "Id": 54,
                "OperationName": "Read",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            },
            {
                "Id": 55,
                "OperationName": "Write",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            }
        ]
    },
    {
        "Id": 24,
        "Name": "Repository.AppMigration",
        "CreatedTimestampUtc": "2019-11-26T09:3
4:40.923Z",
        "ModifiedTimestampUtc": "2019-11-26T09:
34:40.923Z",
        "Operations": [
            {
                "Id": 61,
                "OperationName": "Archive",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 60,
                "OperationName": "Execute",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 58,
                "OperationName": "Read",
                "SecurableTypeId": 24,
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 62,
                "OperationName": "Whatever",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 59,
                "OperationName": "Write",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            }
        ]
    }
]
```

You can also retrieve a specific Securable Type by either its Name or Id.

Direct Consumer API call	C# code using Consumer SDK library
If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/1 If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/Name/InstructionSet Both yield following response: **Return payload** Expand source `{ "Id": 1, "Name": "InstructionSet", "CreatedTimestampUtc": "2019-11-07T13:14:52.75Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.113Z", "Operations": [ {`	Use SecurableTypes object inside the Tachyon connector instance. To retrieve a Securable Type using its Id use: **Retrieving a Securable Type by Id** Expand source `secType = connector.SecurableTypes.Get(1);` To retrieve a Securable Type using its Name use: **Retrieving Securable Type by Name** Expand source

Direct Consumer API call	C# code using Consumer SDK library

<pre>"Id": 2,
"OperationName": "Actioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 4,
"OperationName": "Approver",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 3,
"OperationName": "Questioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 1,
"OperationName": "Viewer",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
}
]
}</pre>
	<pre>secType = connector.Secu rableTypes.Get("Instruct ionSet");</pre>
	"secType" object will contain the same data you can see in the JSON response on the left.

2.5.9.3.4 Getting Applicable Operations

Applicable Operations work in the context of a Securable Type, which is why Tachyon allows you to retrieve all Applicable Operations for given Securable Type. You can use either Name or Id to specify which Securable Type you want Applicable Operations for.

Direct Consumer API call	C# code using Consumer SDK library
If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableTypeId/1 If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableType Name/InstructionSet Both yield the following response:	Use ApplicableOperations object inside the Tachyon connector instance. To retrieve Applicable Operations for a Securable Type using the type's Id use: **Retrieve Applicable Operations using Securable Type's Id** Expand source <pre>operations = connector.Applicable Operations.Get(1);</pre>

Direct Consumer API call	C# code using Consumer SDK library
Return payload Expand source `[{ "Id": 2, "OperationName": "Actioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 4, "OperationName": "Approver", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 3, "OperationName": "Questioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 1, "OperationName": "Viewer", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }]`	To retrieve Applicable Operations for a Securable Type using the type's Name use: **Retrieve Applicable Operations using Securable Type's Name** Expand source `operations = connector.ApplicableOperations.Get("InstructionSet");` "operations" object will contain the same data you can see in the JSON response on the left.

2.5.9.3.5 Getting Permissions

You can retrieve permissions in several different ways, for example for a Principal, a Role or a Securable Type.

Some APIs perform checks for the calling user by pulling the user information from the HTTP request itself. Other APIs allow you to specify which object you want to get permissions for.

2.5.9.3.5.1 Getting permissions for a Principal

Getting all Permissions

Retrieving a principal's permissions is done using the account name (for example "somedomain\jane.doe") of that principal. When directly using the API, you have to encode that account name into base64 before sending it. C# Consumer API SDK will do the encoding for you.

In general, any GET endpoint requires you to base64 encode the principal name, due to the fact that principal names can contain characters that are not allowed in URIs.

When you request permissions for a specific Principal, you'll get the permissions that Principal has stemming from any of their roles.

The following examples use the "somedomain\jane.doe" account.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ== will yield following response: **Return payload** Expand source `[{ "SecurableId": null, "SecurableName": null, "SecurableTypeId": 4, "SecurableTypeName": "Instrumentation", "RoleId": 4, "RoleName": "Infrastructure Administrators", "Allowed": true, "Operations": [{ "PermissionId": 19, "OperationId": 11, "OperationName": "Read", "CreatedTimestampUtc": "2019-06-07T15:12:40.757Z", "ModifiedTimestampUtc": "2019-06-07T15:12:40.757Z" }] }, { "SecurableId": null, "SecurableName": null, "SecurableTypeId": 12, "SecurableTypeName": "ProcessLog", "RoleId": 16, "RoleName": "Log Viewers", "Allowed": true, "Operations": [{ "PermissionId": 61, "OperationId": 33, "OperationName": "Read", "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z", "ModifiedTimestampUtc": "2019-06-07T15:12:`	Use Permissions object inside the Tachyon connector instance. **Retrieving all Permissions for a specific Principal** Expand source `permissions = connector.Permissions.GetForPrincipal("somedomain\\jane.doe");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 13,
    "SecurableTypeName": "SynchronizationLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 62,
        "OperationId": 34,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 14,
    "SecurableTypeName": "Component",
    "RoleId": 17,
    "RoleName": "Component Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 64,
        "OperationId": 35,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 16,
    "SecurableTypeName": "InfrastructureLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
```

Direct Consumer API call	C# code using Consumer SDK library

```
    "Operations": [{
        "PermissionId": 63,
        "OperationId": 39,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
}]
```

Checking for a specific Permission

You can also retrieve permissions given a Principal has on a particular Securable Type.

In the example below, we'll look at InstructionSet related permissions "somedomain\jane.doe" account has.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ==/Type/InstructionSet will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving InstructionSet related permissions for a Principal** Expand source `permissions = connector.Permissions.GetForPrincipalAndType("somedomain\\jane.doe", "InstructionSet");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1,
        "RoleName": "Global Administrators",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
```

```
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

If given Securable Type supports Instances, you can also check a Principal's permissions on a specific instance.

In the example below we'll use a different account called "somedomain\john.doe" and check what permissions it has on Instruction Set with the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/Type/InstructionSet/1 will yield following response: **Return payload** Expand source `[{ "SecurableId": 1, "SecurableName": "MySet", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTimestampUtc": "2020-01-02T12:04:04.963Z" }] }]`	Use Permissions object inside the Tachyon connector instance. **Retrieving Principal's permissions to a specific instruction set** Expand source `permissions = connector.Permissions.GetForPrincipalAndTypeAndInstance("somedomain\\john.doe", "InstructionSet", 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

If the given Principal has no permissions or does not have permissions on given Securable Type and/or instance, an empty collection is returned.

2.5.9.3.5.2 Getting permissions for a Role

You can retieve Role's permissions by using the role Id.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16 will yield following response:
 	
Return payload Expand source
 	
[
{
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 12,
"SecurableTypeName": "ProcessLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [
{
"PermissionId": 61,
"OperationId": 33,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-11-07T13:14:56.673Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.673Z"
}
]
},
{
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 13,
"SecurableTypeName": "SynchronizationLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [
{
"PermissionId": 62,
"OperationId": 34,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-11-07T13:14:56.673Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.673Z"
}
]
	Use Permissions object inside the Tachyon connector instance.
	
	Retrieving all permissions assigned to a Role Expand source
	
	`permissions = connector.Permissions.GetForRole(16);`
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 16,
        "SecurableTypeName": "Infrastructur
eLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 63,
                "OperationId": 39,
                "OperationName": "Read",
                "CreatedTimestampUtc": "201
9-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc": "20
19-11-07T13:14:56.673Z"
            }
        ]
    }
]
```

You can also retrieve all permissions given Role has on a specific Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16/Type/ProcessLog will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions that a Role has on a given Securable Type** Expand source `permissions = connector.Permissions.GetForRoleAndType(16, "ProcessLog");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 12,
        "SecurableTypeName": "ProcessLog
",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "PermissionId": 61,
                "OperationId": 33,
                "OperationName": "Read",
                "CreatedTimestampUtc": "
2019-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc":
"2019-11-07T13:14:56.673Z"
            }
        ]
    }
]
```

And for Securable Types that support Instances, you can also check Permissions on a specific Instance.

Here we'll get permissions for a custom Role that's been assigned a 'Viewer' permissions on an Instruction Set.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/30/Type/InstructionSet/1 will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions a Role has on a specific Instance of a Securable Type** Expand source `permissions = connector.Permissions.GetForRoleAndTypeAndInstance(30, "InstructionSet", 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": 1,
        "SecurableName": "Wszystko",
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instructi
onSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewe
r",
                "CreatedTimestampUtc":
"2020-01-02T12:04:04.963Z",
                "ModifiedTimestampUtc":
"2020-01-02T12:04:04.963Z"
            }
        ]
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

2.5.9.3.5.3 Getting permissions for a Securable Type

You can retrieve permissions granted on a Securable Type using the type's Id. This will return a collection of all permissions on a given type assigned to any of the Roles.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Securable/1 will yield following response:

Return payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 1,
        "RoleName": "Global Administrator
s",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
```

Use Permissions object inside the Tachyon connector instance.

Retrieving all Permisisons on a given Securable Type Expand source

```
permissions = connector.Permission
s.GetForSecurableType(1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
```

```
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 6,
        "RoleName": "Global Questioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 21,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 7,
        "RoleName": "Global Actioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 22,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 8,
```

Direct Consumer API call	C# code using Consumer SDK library


```
        "RoleName": "Global Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 23,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 27,
        "RoleName": "ServiceNow ITSM Conn
ect",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 136,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:18:05.767Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:18:05.767Z"
            }
        ]
    }
]
```


You can also retrieve permissions granted on a specific Instance of a Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Perm	Use Permissions object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

issions/Securable/1/1 will yield following response:
 	
Return payload Expand source
 	
[
{
"SecurableId": 1,
"SecurableName": null,
"SecurableTypeId": 1,
"SecurableTypeName": "Instr uctionSet",
"RoleId": 30,
"RoleName": "MySet Viewers ",
"Allowed": true,
"Operations": [
{
"PermissionId": 13 7,
"OperationId": 1,
"OperationName": "V iewer",
"CreatedTimestampUt c": "2020-01-02T12:04:04.963Z",
"ModifiedTimestampU tc": "2020-01-02T12:04:04.963Z"
}
]
}
]
	Retrieving permissions assigned on a specific inst ance of a Securable Type Expand source
	
	`connector.Permissions.GetForSecurableTyp eAndInstance(1, 1);`
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

2.5.9.3.5.4 Helper APIs

Consumer API has a number of endpoints that are not directly involved with its RBAC system but instead provide various utility functions around the general principal and permission area.

Who am I?

You can ask the API to return the information about the user who's making the request. The API examines the incoming request, retrieves the name of the user from it, makes sure the user is authenticated, and returns information summary about the user, like user's principal name, SID, display name and email.

This endpoint is useful for systems that cannot assume they're running under a particular account, like a browser or an application running in the context of launching user. In those cases this endpoint can be used to obtain the name of the user, which is then fed to the Permissions endpoint seen above in order to retrieve the permissions of the caller.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/whoami will yield following response: **Return payload** Expand source `{ "PrincipalName": "Somedomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-19970", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "Photo": null }`	Use PrincipalSearch object inside the Tachyon connector instance. **Who am I?** Expand source `user = connector.PrincipalSearch.GetCurrentlyLoggedInUser();` "user" object will contain the same data you can see in the JSON response on the left.

Active Directory search

Because in order to add a Principal to Tachyon you have to know the SID of the user, Tachyon's Consumer API exposes endpoints that help you look up users in Active Directory and retrieve information about them, including their SID.

You can perform a search for Active Directory accounts whose common name or sAMAccountName contain given phrase. Details of what exact search is performed are beyond the scope of this page and depend on type of objects being searched for, like users, groups and computers and whether your Tachyon installation is using GC or LDAP.

In the examples below we'll be looking for accounts that contain the phrase "Administrator". How you perform this search depends on your version of Tachyon.

Tachyon versions up to and including 4.1

You simply provide a base64 encoded search string and issue a GET request.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/QWRtaW5pc3RyYXRvcg== will yield following response: **Return payload** Expand source `[ { "PrincipalName": "SomeDomain\\CSU`	Use PrincipalSearch object inside the Tachyon connector instance. **Searching for principals in Active Directory** Expand source

Direct Consumer API call	C# code using Consumer SDK library

<pre>serAdministrator",
"ExternalId": "S-1-5-21-1202660629-789336058-1353024091-8145",
"Email": null,
"DisplayName": "CSUserAdministrator",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\CSVoiceAdministrator",
"ExternalId": "S-1-5-21-1202660629-789336058-1343023091-8024",
"Email": null,
"DisplayName": "CSVoiceAdministrator",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exchange Organization Administrators",
"ExternalId": "S-1-5-21-1202660629-789336058-1343324091-6881",
"Email": null,
"DisplayName": "Exchange Organization Administrators",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exchange Public Folder Administrators",
"ExternalId": "S-1-5-21-1202660629-789336058-1340024091-7747",
"Email": null,
"DisplayName": "Exchange Public Folder Administrators",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exchange Recipient Administrators",
"ExternalId": "S-1-5-21-1202660629-789336058-1340024091-6913",
"Email": null,
"DisplayName": "Exchange Recipient Administrators",
"IsGroup": true</pre>
	<pre>results = connector.PrincipalSearch.SearchForPrincipals("Administrator");</pre>
	"results" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
        },
        {
                "PrincipalName": "SomeDomain\\Exc
hange View-Only Administrators",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange View-Onl
y Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Sec
urity Administrator",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security Administ
rator",
                "IsGroup": true
        }
]
```

Tachyon versions 5.0 and newer

Here you have greater control over the search. A POST request should be used and in its payload you can specify, apart from the search string, number of results returned, sort column and order.

You can search for "user" and "group" object types and sort in either ascending (using "ASC") or descending order (using "DESC") on following columns: "cn", "mail", "sAMAccountName", "description", "objectSid", "displayName". Column names, object types and sorting direction string are not case sensitive.

You have to provide the SearchText (the text to search for) and at least one object type. Pagesize and sort are optional and if not provided will default to respectively 100 and ascending sorting on display name.

One things that you have to keep in mind when it comes to PageSize property is that, due to internal implementation, it defines the size of the page returned by Active Directory. This means that you can get fewer results than you requested, even if there would have been more to return because Tachyon filters out users already in the system. As an example, if you define the page size to be 10 and the Active Directory has 18 entries that match your search string, it will return 10 entries. Now let us assume that out of those 10 entries 1 user is already in Tachyon. This user will be filtered out of the return data set, so you will receive only 9 entries.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalSearch with following payload:	Use PrincipalSearch object inside the Tachyon connector instance.

Direct Consumer API call

Request payload Expand source

```
{
        "SearchText": "Administrator",
        "ObjectTypes": ["user", "group
"],
        "PageSize": 100,
        "Sort": {
                "Column": "displayname",
                "Direction": "ASC"
        }
}
```

will yield following response:

Return payload Expand source

```
[
        {
                "PrincipalName": "SomeDo
main\\CSUserAdministrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1353024091-8145",
                "Email": null,
                "DisplayName": "CSUserAd
ministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\CSVoiceAdministrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343023091-8024",
                "Email": null,
                "DisplayName": "CSVoiceA
dministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Organization Administrato
rs",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343324091-6881",
                "Email": null,
                "DisplayName": "Exchange
Organization Administrators",
                "IsGroup": true
```

C# code using Consumer SDK library

Retrieving 100 entries sorted by displayna me Expand source

```
var searchParams = new ActiveDirect
orySearchModel
{
        SearchText = "Administrator
",
        ObjectTypes = new List<stri
ng> { "user", "group" },
        PageSize = 100,
        Sort = new SortSpec
        {
                Column = "displayna
me",
                Direction = "ASC"
        }
};

results = connector.PrincipalSearc
h.SearchForPrincipals(searchParam
s);
```

"results" object will contain the same data you can see in the JSON response on the left.

```
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Public Folder Administrat
ors",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-7747",
                "Email": null,
                "DisplayName": "Exchange
Public Folder Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Recipient Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange
Recipient Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange View-Only Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange
View-Only Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Security Administrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security
Administrator",
                "IsGroup": true
        }
]
```

2.5.9.3.5.5 Retrieving Active Directory information about a specific account

You can also retrieve user's information by their principal name, though this endpoint only returns authenticated users who are also Principals in Tachyon (either directly or throught group membership).

This means that in order to obtain information about somedomain\jane.doe using this endpoint, somedomain\jane.doe has to be a domain account that's either a Principal in Tachyon, or belongs to an Active Directory group that is a Principal in Tachyon.

When using the API directly, the principal name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/user/c29tZWRvbWFpblxqYW5lLmRvZQ==will yield following response: **Return payload** Expand source `[ { "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "jane.doe@somedomain.com", "DisplayName": "Jane Doe", "IsGroup": false } ]`	Use PrincipalSearch object inside the Tachyon connector instance. **Retrieve information about a principal from Active Directory** Expand source `results = connector.PrincipalSearch.SearchForUser("SomeDomain\\Jane.Doe");` "results" object will contain the same data you can see in the JSON response on the left.

Retrieving members of an Active Directory group

You can ask Tachyon to retrieve members of an Active Directory group. This group does not have to be a principal in Tachyon for this functionality to work.

When using the API directly the group name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

In the example below we'll use an AD group called "Tachyon users" and retrieve its members.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/GetMembers/VGFjaHlvbiB1c2Vycw==will yield following response:	Use PrincipalSearch object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

Return payload Expand source
 	
[
{
"PrincipalName": "SomeDomain\\Jane.Doe",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842",
"Email": "jane.doe@somedomain.com",
"DisplayName": "Jane Doe",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Meetra.Surnik",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-8631",
"Email": "Meetra.Surnik@somedomain.com",
"DisplayName": "Meetra Surnik",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Keiran.Halcyon",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-1138",
"Email": "Keiran.Halcyon@somedomain.com",
"DisplayName": "Keiran Halcyon",
"IsGroup": false
}
]
	Retrieving members of an Aciv e Directory group Expand source
	
	`results = connector.PrincipalSearch.GetGroupMembers("Tachyon users");`
	
	"results" object will contain the same data you can see in the JSON response on the left.

2.5.9.4 Configuring Tachyon RBAC through the Consumer API

In this section we will look at configuring RBAC emphasizing operations on Principals, Roles and Permissions but also covering Securable Types and Operations later on.

2.5.9.4.1

2.5.9.4.1.1 Adding Principals from Active Directory

Adding Principals to Tachyon will be one of the first things you'll do after installing the system. A fresh installation of Tachyon (which in this context means an installation where there was no previous Master database or such database was dropped) will have two Principals - the account used to install the Tachyon Server (and created the database to be specific) and NT AUTHORITY\Network Service account.

These accounts have limited permissions so to properly use the system you should add more Principals. In order to add a principal to Tachyon you will need an External Id, also called SID, for an active directory account that you wish to add as a principal. Although you can obtain account details through various means. In Active Directory search we've seen Tachyon exposes endpoints that return this information and enables you to search for Active Directory accounts.

Here we'll assume you already have the account details and focus on adding an account as a Principal to Tachyon.

In the example below we'll "SomeDomain\Jane.Doe" account and add it to Tachyon.

To add an account as a Principal you'll at the very least need to supply an ExternalId (SID) and Principal Name, though its advised to also supply a Display Name. Also, unless you set the Enabled flag to true, the newly created account will be disabled by default.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Principals with following payload:
 	
Request payload Expand source
 	
`{ "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "IsGroup": false, "Enabled": true }`
 	
Will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": false, "DisplayName": "Jane Doe",`
	Use Principals object inside the Tachyon connector instance.
	
	Adding a Principal to Tachyon Expand source
	
	`var payload = new Principal { PrincipalName = "SomeDomain\\Jane.Doe", ExternalId = "S-1-5-21-1202660629-789336058-1343024091-23842", Email = "Jane.Doe@SomeDomain.com", DisplayName = "Jane Doe", IsGroup = false, Enabled = true };`
	
	`principal = connector.Principals.Add(payload);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "IsGroup": false
}
```

Note

You cannot add a Principal with the same PrincipalName or ExternalId as another Principal that's already in the system, because both of those values must be unique.

Updating a Principal

Updating a Principal is very similar to adding one. You have to use a PUT verb instead of a POST if using the API directly, or Update method instead of Add if you're using the SDK. You also have to provide an Id that belongs to an existing Principal. Remember, you cannot modify system Principals.

Again, you have to remember that ExternalId and PrincipalName must remain unique so you cannot change Principal's PrincipalName or ExternalId to one that matches another Principal's.

2.5.9.4.2 Configuring Roles

Roles are the pivotal point of an RBAC system.

Tachyon comes with a number of pre-defined roles, which should be sufficient to start with, but in time you will most likely create custom roles, either to restrict access to instruction sets or to cover sets of permission required by specific roles within your organization.

Adding, Editing and Removing a Role

First let's look at creating a Role. This can be done either in isolation, where just the role is created and any permissions must be assigned in subsequent call(s), or as a more complete package, where a Role is created along with a set of Permissions, which can include Management Groups.

Regardless of which approach is used, Principals have to be assigned separately.

To create a Role all you require is a name, and this name must be unique. Description can optionally be provided and you can't create system roles.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roleswith following payload: **Request payload** Expand source `{ "Name": "Custom role", "Description": "this is a description" }` Will yield following response:	Use Roles object inside the Tachyon connector instance. **Creating a Role** Expand source `var payload = new Role { Name = "Custom role", Description = "this is a description" };`

Direct Consumer API call	C# code using Consumer SDK library
Return payload Expand source `{ "Id": 31, "Name": "Custom role", "Description": "this is a description", "CreatedTimestampUtc": "2020-01-21T13:36:55.8956916Z", "ModifiedTimestampUtc": "2020-01-21T13:36:55.8956916Z", "SystemRole": false }`	`role = connector.Roles.Add(payload);` "role" object will contain the same data you can see in the JSON response on the left.

Creating a Role with Permissions is possible by joining Role creation we've just seen with creating permissions, which is described a bit later.

This effectively means that we take a regular Role creation payload and add a 'Permissions' property to it. That propery is an array of permissions, which we are yet to look at in greater details.

The example below creates a role, which has "Viewer" and "Approver" permissions on Instruction Set with the Id of 1 and "Actioner", "Questioner", "Viewer" and "Approver" permissions on Instruction Set with the Id of 2. It also has permissions on three Management Groups.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Completewith following payload: **Request payload** Expand source `{ "Name": "Complete Role 1", "Description": "This is a test role", "Permissions": [{ "SecurableId": 2, "SecurableTypeId": 1, "Allowed": true, "Operations": [{ "OperationId": 2 }, { "OperationId": 4 }, { "OperationId": 3`	Use Roles object inside the Tachyon connector instance. **Creating a Role with Permissions** Expand source `var payload = new Tachyon.SDK.Consumer.Models.Send.RoleWithPermissionsAndManagementGroups { Name = "Complete Role 1", Description = "This is a test role", Permissions = new AggregatedPermission[] { new AggregatedPermission {`

Direct Consumer API call

```
        },
        {
            "OperationId": 1
        }]
    },
    {
        "SecurableId": 1,
        "SecurableTypeId": 1,
        "Allowed": true,
        "Operations": [{
            "OperationId": 1
        },
        {
            "OperationId": 4
        }]
    }],
    "ManagementGroupIds": [10,11,12]
}
```

Will yield following response:

Return payload Expand source

```
{
    "Permissions": [
        {
            "SecurableId": 1,
            "SecurableName": null,
            "SecurableTypeId": 1,
            "SecurableTypeName": "InstructionSet
",
            "RoleId": 32,
            "RoleName": "Complete Role 1",
            "Allowed": true,
            "Operations": [
                {
                    "PermissionId": 142,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 143,
                    "OperationId": 4,
```

C# code using Consumer SDK library

```
            SecurableId =
2,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 2 },
                new Permiss
ionOperation { OperationId
= 3 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        },
        new AggregatedPermi
ssion
        {
            SecurableId =
1,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        }
    },
    ManagementGroupIds = ne
w List<int> { 10, 11, 12 }
};
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
}
]
},
{
"SsecurableId": 2,
"SecurableName": null,
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet",
"RoleId": 32,
"RoleName": "Complete Role 1",
"Allowed": true,
"Operations": [
{
"PermissionId": 138,
"OperationId": 2,
"OperationName": "Actioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 139,
"OperationId": 4,
"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 140,
"OperationId": 3,
"OperationName": "Questioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{</pre>
	<pre>role = connector.Roles.Add(payload);</pre>
	"role" object will contain the same data you can see in the JSON response on the left.

```
                    "PermissionId": 141,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                }
            ]
        }
    ],
    "ManagementGroups": [
        {
            "Id": 10,
            "Name": "Management group number 9 f
rom repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 9 from repositor
y Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 1,
            "UsableId": "9",
            "HashOfMembers": "4CFD42DAA10FD0FDE0
B4E6BE1116155AC6180EE4E9ADF543F39E97D9B8C51B25",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.677Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.343Z"
        },
        {
            "Id": 11,
            "Name": "Management group number 10
from repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 10 from reposito
ry Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 10,
            "UsableId": "10",
            "HashOfMembers": "F51DE677220883E4A6
84DBB7D965110DE74517D714B3459DA4A98C3B56C58D1C",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.68Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.43Z"
```

```
        },
        {
            "Id": 12,
            "Name": "Management group number 11
from repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 11 from reposito
ry Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 27,
            "UsableId": "11",
            "HashOfMembers": "5D25CC2DEBF1D76B81
602884623FDE878FE8EF57568A6F97A9F04242FEF2655D",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.68Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.517Z"
        }
    ],
    "Id": 32,
    "Name": "Complete Role 1",
    "Description": "This is a test role",
    "CreatedTimestampUtc": "2020-01-22T07:18:57.
7082854Z",
    "ModifiedTimestampUtc": "2020-01-22T07:18:5
7.7082854Z",
    "SystemRole": false
}
```


To update a Role you should use PUT versions of the two endpoints seen above and provide one additional property in the JSON payload - the Id of the Role being modified. If you're using the Consumer SDK, you should call Update method instead of Add and you must provide an Id in the payload object sent to the API.

Since the names of Roles have to be unique, you won't be able to change the name of a Role to one that is the same as another Role.

Changing the Role details with https://my.tachyon.server/Consumer/Roles (or using the Update method in Consumer SDK that takes Role object type) will change just the details of the role will not change the permissions in any way. It will not change which Mangement Groups or Principals are assigned to the Role either.

Changing the Role via https://my.tachyon.server/Consumer/Roles/Complete will not change which Principals are assigned to the Role, but it will change the Permissions and Management Groups. When using this endpoint with PUT verb (or using the Update method in Consumer SDK that takes RoleWithPermissionsAndManagementGroups object type), all of the Permissions for given Role will be replaced with Permissions provided in the payload and all Management Groups will be replaced with Management Groups provided in the payload. This means that if you do not provide Permissions (omit the property, provide a

null value or an empty array), all Permissions will be removed from the Role. Likewise, if you do not provide ManagementGroups all Management Groups will be unassigned from the Role.

Deleting a role is straightforward and you just need its Id. You can also delete multiple Roles in one call by providing a collection of Role Ids. In either case, you cannot delete a system Role.

Direct Consumer API call	C# code using Consumer SDK library

To delete a single Role make a DELETE request to https://my.tachyon.server/Consumer/Roles/31 ,which will delete the Role with the Id of 31.
 	
To delete multiple Roles, make a DELETE request and send an array of Ids to https://my.tachyon.server/Consumer/Roles. To delete Roles with Ids 31, 32, 33 and 34 you would send this payload:
 	
Request payload Expand source
 	
`[31,32,33,34]`
	Use Roles object inside the Tachyon connector instance.
	
	To delete a single Role:
	
	Deleting the Role with Id 31 Expand source
	
	`connector.Roles.Delete(31);`
	
	To delete multiple Roles:code
	
	Deleting Roles with Ids 31,32,33 and 34 . Expand source
	
	`connector.Roles.DeleteMultiple(new List<int> {31, 32, 33, 34});`

Assigning and unassigning Principals to Roles

As we've seen above when we covered retrieving Principals and Roles, the association between Principals and Roles can be examined from either end: which Principals are assigned to a given Role or which Roles given Principal has assigned.

Looking at the issue from the perspective of a Principal, you can assign or unassign Roles(s) to a Principal, or replace any Roles that Principal has with a given set of Roles.

First, let's pick a Principal that we will assign Roles to. In the example below I'll use "SomeDomain\\Jane.Doe", which has the Id of 3 and I'll assign 4 Roles to that Principal: Instruction Set Administrators, Permissions Administrators, Infrastructure Administrators, Global Questioners which have respective Ids of 2, 3, 4 and 6.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Request payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source `[2,3,4,6]`	Use PrincipalRoles object inside the Tachyon connector instance. **Adding multiple Roles to a Principal** Expand source `roles = connector.PrincipalRoles.AddRolesToPri`

Direct Consumer API call	C# code using Consumer SDK library

will yield following response:
 	
Return payload Expand source
 	
[
{
"PrincipalId": 3,
"RoleId": 2,
"CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
"Role": {
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 2,
"Id": 2,
"Name": "Instruction Set Administrators",
"Description": "This role can add product packs; add, modify and delete instruction sets; and delete instruction definitions",
"CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
"SystemRole": true
},
"Principal": null
},
{
"PrincipalId": 3,
"RoleId": 3,
"CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
"Role": {
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 2,
"Id": 3,
"Name": "Permissions Administrators",
"Description": "This role can add or remove users; view all roles; add, modify and delete custom roles; assign roles to any instruction sets and define their permissions; and view the admin log",
"CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
	ncipal(3, new List<int> { 2, 3, 4, 6 });
	
	"roles" object will contain the same data you can see in the JSON response on the left.

```
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.117Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 4,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.
31Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned":
false,
            "AssignedPrincipalCount": 1,
            "Id": 4,
            "Name": "Infrastructure Administrators",
            "Description": "This role can view syste
m status instrumentation and system information",
            "CreatedTimestampUtc": "2019-11-07T13:1
4:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.493Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 6,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.
31Z",
        "Role": {
            "AssignedManagementGroupCount": 1,
            "HasAllDevicesManagementGroupAssigned":
true,
            "AssignedPrincipalCount": 1,
            "Id": 6,
            "Name": "Global Questioners",
            "Description": "This role can ask questi
ons and view responses for all instruction sets",
            "CreatedTimestampUtc": "2019-11-07T13:1
4:54.957Z",
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.12Z",
            "SystemRole": true
```

Direct Consumer API call	C# code using Consumer SDK library

```
        },
        "Principal": null
    }
]
```

Now let's replace the Roles for that Principal with 2 other Roles: Custom Properties Administrators and Consumer Administrators, which have Ids 9 and 10 respectively.

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload:

Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source

```
[9,10]
```

will yield following response:

Return payload Expand source

```
[
    {
        "PrincipalId": 3,
        "RoleId": 9,
        "CreatedTimestampUtc": "2020-01-23T0
8:24:25.36Z",
        "Role": {
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAssi
gned": false,
            "AssignedPrincipalCount": 2,
            "Id": 9,
            "Name": "Custom Properties Admini
strators",
            "Description": "This role can ad
d, edit or delete custom properties",
            "CreatedTimestampUtc": "2019-11-0
7T13:14:55.133Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:55.133Z",
            "SystemRole": true
        },
```

Use PrincipalRoles object inside the Tachyon connector instance.

Replacing all Roles for a Principal with given Roles Expand source

```
roles = connector.PrincipalRoles.ReplaceRolesForPrincipal(3, new List<int> { 9, 10 });
```

"roles" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library


```
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T0
8:24:25.36Z",
        "Role": {
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAssi
gned": false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators
",
            "Description": "This role can man
age Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-0
7T13:14:55.523Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```


Lastly, I'll remove the assiciation to Custom Properties Administrators Role from the Principal

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source `[9]` will yield following response: **Response payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Deleting multiple Roles from a Principal** Expand source `connector.PrincipalRoles.DeleteRolesFromPrincipal(3, new List<int> { 9 });` The method will return a boolean value.

Direct Consumer API call	C# code using Consumer SDK library

```
[
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T08:24:
25.36Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned
": false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators",
            "Description": "This role can manage
Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-07T1
3:14:55.523Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```

Looking at the issue from the perspective of a Role, you can assign or unassign Principal(s) to a Role, or replace any Principals that Role has with a given set of Principals.

In the examples below we'll work with the Custom Role we've created above. It has the Id of 31 and first we'll assign 3 Principals to it:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source `[3, 5, 6]` will yield following response: **Response payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Adding multiple Principals to a Role** Expand source `principals = connector.PrincipalRoles.AddPrincipalsToRole(31, new List<int> { 3, 5, 6 });`

Direct Consumer API call	C# code using Consumer SDK library

<pre>[
    {
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-1202660629-789
336058-1343024091-23842",
            "PrincipalName": "SomeDomain\\Jane.Doe
",
            "Email": "Jane.Doe@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Jane Doe",
            "IsGroup": false
        }
    },
    {
        "PrincipalId": 5,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 1,
            "ExternalId": "S-1-5-21-1202660629-789
336158-1349024092-24850",
            "PrincipalName": "SomeDomain\\Alastor.
Rushal",
            "Email": "Alastor.Rushal@SomeDomain.co
m",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Alastor Rushal",
            "IsGroup": false
        }</pre>
	"principals" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    },
    {
        "PrincipalId": 6,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 2,
            "ExternalId": "S-1-5-21-1202660629-789
306118-1349024091-24337",
            "PrincipalName": "SomeDomain\\Marius.G
age",
            "Email": "Marius.Gage@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:56.687Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:56.687Z",
            "SystemPrincipal": false,
            "DisplayName": "Marius Gage",
            "IsGroup": false
        }
    }
]
```

Now we will replace the existing 3 Principals with 2, one of which, "SomeDomain\\Jane.Doe" was already assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31
 	
Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source
 	
`[3, 4]`
 	
will yield following response:
 	
Return payload Expand source
 	
`[`
`  {`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Replacing all Principals for a Role with given Principals Expand source
	
	`principals = connector.PrincipalRoles.ReplacePrincipalsForRole(31, new List<int> { 3, 4 });`
	
	"principals" object will contain the same data you can see in the JSON response on the left.

```
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23
T10:00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-120266
0629-789336058-1343024091-23842",
            "PrincipalName": "SomeDomain\\
Jane.Doe",
            "Email": "Jane.Doe@SomeDomain.
com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-1
1-07T13:14:52.777Z",
            "ModifiedTimestampUtc": "2019-
11-07T13:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Jane Doe",
            "IsGroup": false
        }
    },
    {
        "PrincipalId": 4,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23
T10:00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-327632
6578-728399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\
John.Doe",
            "Email": "John.Doe@SomeDomain
",
            "Enabled": true,
            "CreatedTimestampUtc": "2016-1
1-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-
11-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "John Doe",
            "IsGroup": false
        }
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

And finally we'll remove all "SomeDomain\\John.Doe" from the Role, leaving just "SomeDomain\\Jane.Doe" assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library

Making a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31
 	
Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source
 	
`[4]`
 	
will yield following response:
 	
Return payload Expand source
 	
`[{ "PrincipalId": 3, "RoleId": 31, "CreatedTimestampUtc": "2020-01-23T10:00:11.63Z", "Role": null, "Principal": { "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": false, "DisplayName": "Jane Doe", "IsGroup": false } }]`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Removing multiple Principals from a Role Expand source
	
	`connector.PrincipalRoles.DeletePrincipalsFromRole(31, new List<int> { 4 });`
	
	The method will return a boolean value.

As we can see, the Principal and Role assignment, unassignment and replacement endpoints mirror each other covering the approach from either Principal or Role point of view.

Lastly, you can create or remove a link between a single Principal and a single Role.

In this example we'll create a link between Customer Role, which has the id of 31 and "SomeDomain\\John.Doe" Principal, which has the Id of 4.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles
 	
Request payload sent to https://my.tachyon.server/Consumer/PrincipalRoles Expand source
 	
`{ "PrincipalId": 4, "RoleId": 31 }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "PrincipalId": 4, "RoleId": 31, "CreatedTimestampUtc": "2020-01-23T10:41:24.503Z", "Role": { "AssignedManagementGroupCount": 0, "HasAllDevicesManagementGroupAssigned": false, "AssignedPrincipalCount": 4, "Id": 31, "Name": "Custom role", "Description": "this is a description", "CreatedTimestampUtc": "2020-01-21T13:36:55.897Z", "ModifiedTimestampUtc": "2020-01-21T13:36:55.897Z", "SystemRole": false }, "Principal": { "Id": 4, "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1009", "PrincipalName": "SomeDomain\\John.Doe", "Email": "John.Doe@SomeDomain", "Enabled": true, "CreatedTimestampUtc": "2016-11-30T15:10:10.73`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Assigning Principal to a Role Expand source
	
	`var payload = new PrincipalRole { PrincipalId = 4, RoleId = 31 }; principal = connector.PrincipalRoles.Add(payload);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
Z",
        "ModifiedTimestampUtc": "2016-11-30T15:10:10.7
3Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
}
```

and now we'll remove the link we've just created:

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/Role/31/Principal/4 will not yield any response payload so you should check the HTTP return code to see if the operation was successful.	Use PrincipalRoles object inside the Tachyon connector instance. **Unassigning Principal from a Role** Expand source `connector.PrincipalRoles.Delete(31, 4);` The method will return a boolean value.

Adding and removing Permissions to a Role The main method of assigning Permissions is to send two collections, one with Permissions that need to be changed or updated and another one with permissions that should be removed.

Crafting the correct payload should be done carefully as omitting an entry can result in deletion of an existing Permission.

PermissionsToSaveOrUpdate collection should have permissions you either want to add or modify. You only need to provide Ids, names are ignored. It is important to understand when the system considers given operation an 'updated' and when a 'creation' as it will behave slightly differently.

Because a Permission is between a Securable Type and a Role, with possible addition of Securable Id, those three properties define it. Operations are considered a property of a Permission and they are always considered together - i.e. a single Permission can Allow or Deny a set of Operations, but it cannot mix and match.

When you request a Permission to be added or updated, Consumer API will look at the SecurableId, SecurableTypeId and RoleId properties provided and check for any permissions already existing for this combination of values. If none are found, a new permission is added with the operations specified in the payload. If an entry is found, then Operations are examined next. If the existing Permission has Operations that aren't present in the Permission sent to the API, they are removed. Any operations present in the Permission sent to the API that do not exist in the Permission that is already in the system are added.

As you might have noticed, PermissionsToSaveOrUpdate can be used to remove Permissions by omitting Operations you want to remove. Furthermore, if you provide a Permission without any Operations, the entire Permission is removed.

Lets look at an example that should help us better understand how this functionality works.

Let us assume that we have a Permission on Instruction Set "My set" assigned to Role "Custom Role" with just one Operation "View". "My Set" has the Id of 4, "Custom Role" has the Id of 31, "Instruction Set" Securable Type has the Id of 1 and "Viewer" has the Id of 1. This means that the existing Permission will look as follows:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        }
        ]
}
```

Now let us assume that a request comes into the Consumer API that looks as follows:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 1
                        },
                                {
                                "OperationId": 3
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

This request matches a Permission we already have, because it has the same SecurableId, SecurableTypeId and RoleId, but the Operations are slightly different - Id of 3 has been added, which represents 'Questioner' Operation. Following the rules outlined above, the API will match this to the existing Permission and see that Operation with the Id of 1 already exists, so no action

is needed but the Operation with the Id of 3 doesn't exist, so it needs to be added. Afterwards we will end up with a permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        },
                {
                "OperationId": 3
        }
        ]
}
```

If now we were to make another request with with following payload:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 2
                        },
                                {
                                "OperationId": 4
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

The Consumer API will again recognise that a Permission already exists that matches the one in the payload and will examine the Operations. Because neither operation 1 nor 3, which do exist in the Permission already in the system are present in the payload, they will be removed. Operations 2 and 4 (Actioner and Approve respectively) do not exist in the Permission so they will be added and changed made will result in a Permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 2
        },
                {
                "OperationId": 4
        }
        ]
}
```

As a last example let's look at what will happen when we send this payload to the API:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": []
        }
    ],
    "PermissionsToDelete": []
}
```

Again, the system will recognise this as an existing Permission and will remove Operations 2 and 4 because they do not exist in the request payload. But since there are no Operations to add, the entire Permission will be removed.

Lastly, let's look at how an example API call would look like:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Permissions with following payload:	Use Permissions object inside the Tachyon connector instance.
Payload sent to https://my.tachyon.server/Consumer/Permissions Expand source	**Creating Permissions** Expand source

`{`
`"PermissionsToSaveOrUpdate": [`
`{`
	`var payload = new Permi`
	`ssionsSaveContainer`
	`{`

Direct Consumer API call

```
            "Allowed": true,
            "SecurableTypeId": 5,
            "SecurableId": null,
            "RoleId": "31",
            "Operations": [
                {
                    "OperationId": 12
                },
                {
                    "OperationId": 13
                }
            ]
        }
    ],
    "PermissionsToDelete": []
}
```

will yield following response:

Response payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 5,
        "SecurableTypeName": "CustomProperty",
        "RoleId": 31,
        "RoleName": "Custom Role",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 191,
                "OperationId": 12,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
14:47:13.023Z"
            },
            {
                "PermissionId": 192,
                "OperationId": 13,
                "OperationName": "Write",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
```

C# code using Consumer SDK library

```
    PermissionsToSaveOr
Update = new List<Aggre
gatedPermission>
    {
        new AggregatedP
ermission
        {
            RoleId = 3
1,
            SecurableTy
peId = 5,
            Operations
= new List<PermissionOp
eration>
            {
                new Per
missionOperation
                {
                    Ope
rationId = 12
                },
                new Per
missionOperation
                {
                    Ope
rationId = 13
                }
            }
        }
    }
};
permissions = connecto
r.Permissions.AddOrUpda
te(payload);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
14:47:13.023Z"
            }
        ]
    }
]
```

2.5.9.4.3 Configuring Management Groups

Management Groups can only be assigned to a Role which has Permission to at least one Instruction Set. This prerequisite stems from internal implementation and it has to be taken into account when designing Permissions in your system.

The example below assumes that Role "Custom Role" with the Id of 31 has at least one Instruction Set Permission assigned to it.

You can assign or unassign either a single Management Group or a collection of Management Groups. To assign a single Management Group to a Role you'd issue a POST request to https://my.tachyon.server/Consumer/Roles/[role Id]/ManagementGroups /[management group id], so for instance to assign Management Group wit the Id of 7 to role with the Id of 31 you'd call https://my.tachyon.server/Consumer/Roles/31/ManagementGroups /7.

Making a DELETE request instead of POST will results in the Management Group being unassigned from the Role.

These requests do not require any payload because all information is already present in the URI. Consumer SDK equivalents are Roles.AddPermissionForManagementGroup(...) and Roles.RemovePermissionForManagementGroup(...) respectively, which simply take the Ids of the Role and the Management Group.

To assign or unassign Management Groups in bulk you have to use POST and DELETE respectively tohttps://my.tachyon.server/Consumer/Roles/ManagementGroups, and below is an example of the payload that should be sent with the request.

Direct Consumer API call	C# code using Consumer SDK library

To assign multiple Management Groups to a Role in a single call, make a POST request to https://my.tachyon.server/Consumer/Roles/ManagementGroupswith following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Roles/ManagementGroups Expand source
 	
`{`
`  "RoleId": 31,`
	Use Roles object inside the Tachyon connector instance.
	
	Assigning Management Groups to a Role Expand source
	
	`var payload = new Tachyon.SDK.Consumer.Models.Send.ManagementGroupsForRole`
	`{`
	`  RoleId = 31,`
	`  ManagementGroupIds`

Direct Consumer API call	C# code using Consumer SDK library

```
  "ManagementGroupIds": [4,5]
}
```

which will yield following response:

Return payload Expand source

```
{
    "Role": {
        "Id": 31,
        "Name": "Custom role",
        "Description": "this is a description",
        "CreatedTimestampUtc": "2020-01-21T13:36:
55.897Z",
        "ModifiedTimestampUtc": "2020-01-21T13:3
6:55.897Z",
        "SystemRole": false
    },
    "ManagementGroups": [
        {
            "Id": 4,
            "Name": "Management group number 3 fr
om repository Default Inventory",
            "Description": "This is the descripti
on for Management group number 3 from repository
Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,
            "UsableId": "3",
            "HashOfMembers": "24D13054496E24C9577
4CD8E983B09C2487D60819C627750B1C16FFDEDB23DF9",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.667Z",
            "ModifiedTimestampUtc": "2020-01-08T1
6:31:02.807Z"
        },
        {
            "Id": 5,
            "Name": "Management group number 4 fr
om repository Default Inventory",
            "Description": "This is the descripti
on for Management group number 4 from repository
Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,
```

```
= new List<int>{ 4, 5 }
};
role = connector.Roles.AddP
ermissionForManagementGroup
s(payload);
```

"role" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
            "UsableId": "4",
            "HashOfMembers": "7784ED3BDA9B79F8635
E728E1E0F529200D44F3F7AABAE1A26B2ABA6DB957EDB",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.67Z",
            "ModifiedTimestampUtc": "2020-01-08T1
6:31:02.893Z"
        }
    ]
}
```

Unassigning Management Groups from a role is a mirror operation using the DELETE verb on the same web address:

Direct Consumer API call	C# code using Consumer SDK library

To unassign multiple Management Groups from a Role in a single call, make a DELETE request to https://my.tachyon.server/Consumer/Roles/ManagementGroups with following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Roles/ManagementGroups Expand source
 	
`{`
`  "RoleId": 31,`
`  "ManagementGroupIds": [4]`
`}`
 	
This call will not return any data.
	Use Roles object inside the Tachyon connector instance.
	
	Unassigning Management Groups from a Role Expand source
	
	`var payload = new Tachyon.SDK.Consumer.Models.Send.ManagementGroupsForRole`
	`{`
	`  RoleId = 31,`
	`  ManagementGroupIds = new List<int> { 4 }`
	`};`
	`connector.Roles.RemovePermissionForManagementGroups(payload);`

2.5.9.4.4 Configuring Securable Types and Applicable Operations

We have already seen how to retrieve Securable Types and now we'll look at how we can create and modify them.

All you need to create a Securable Type is a name, which must be unique.

In order to delete a Securable Type, you have to first delete all Permissions that use that Securable Types and all Applicable Operations linked to the Securable Type.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/SecurableTypes
 	
Paylaod sent to https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Name": "Custom securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Custom securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "ModifiedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "Operations": null }`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Creating a new Securable Type Expand source
	
	`var payload = new SecurableType { Name = "Custom securable type" }; return connector.SecurableTypes.Add(payload);`
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

If you wish to update a securable type you'll have to provide its Id:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/SecurableTypes**Paylaod sent to** https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Id": 26, "Name": "Modified securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Modified securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Modifying an existing Securable Type Expand source
	
	`var payload = new SecurableType { Id = 26, Name = "Modified securable type"`

Direct Consumer API call	C# code using Consumer SDK library

",
"ModifiedTimestampUtc": "2020-01-24T14:38:24.8756464Z
",
"Operations": null
}
	};
	return connector.SecurableTypes.Update(payload);
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

and to delete this Sedurable Type:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/SecurableTypes/26 This call will not yield any response.	Use SecurableTypes object inside the Tachyon connector instance. **Deleting a Securable Type** Expand source connector.SecurableTypes.Delete(26);

It is worth noting that Securable Types are created "empty" and that Applicable Operation have to be created separately.

Dealing with Operations

Applicable Operations exist only in the context of a Securable Type. While Applicable Operations must be unique within a single Securable Type, they do not have to be unique between Securable Types.

To create an Operation you have to provide either Id or Name (but not both) of Securable Type for which the Operation should be created. In the example below we'll use the Securable Type we've just created, which has the Id of 26.

Direct Consumer API call	C# code using Consumer SDK library
Make a POST request to https://my.tachyon.server/Consumer/ApplicableOperations to use Securable Type Id	Use ApplicableOperations object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{`
`  "OperationName": "View",`
`  "SecurableTypeId": 26`
`}`
 	
or with this payload to use Securable Type Name:
 	
Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{`
`  "OperationName": "View",`
`  "SecurableTypeName": "Modified securable type"`
`}`
 	
Both will yield following response:
 	
Return payload Expand source
 	
`{`
`  "Id": 63,`
`  "OperationName": "View",`
`  "SecurableTypeId": 26,`
`  "SecurableTypeName": "Modified securable type"`
`}`
	To use Securable Type Id:
	
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation`
	`{`
	`  SecurableTypeId = 26,`
	`  OperationName = "View"`
	`};`
	`operation = connector.ApplicableOperations.Add(payload);`
	
	To use Securable Type Name:
	
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation`
	`{`
	`  SecurableTypeName = "Modified securable type",`
	`  OperationName = "View"`
	`};`
	`operation = connector.ApplicableOperations.Add(payload);`
	
	"operation" object will contain the same data you can see in the JSON response on the left.

To delete an Operation you just need its Id.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/ApplicableOperations/63	Use ApplicableOperations object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
	Deleting an Applicable Operation Expand source `connector.ApplicableOperations.Delete(63);`

2.5.9.5 Introduction

This section contains an overview of Tachyon role based access control (RBAC) and its components and how to configure Tachyon RBAC. This is not intended as an in-depth explanation of what RBAC is, but as a demonstration of how it's been implemented in Tachyon and how users can configure it programmatically.

The C# examples assume you're using Tachyon Consumer SDK and you have an instantiated an instance of Tachyon connector class in an object called connector. All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API.

In the following examples this detailis leftout, stating that the returned object contains the data.For example, when we say that XYZ object contains certain data, this means the ReceivedObject property inside said object contains that data.

2.5.9.6 Basics of role based access control (RBAC)

Role-based access control is an access control mechanism defined around roles and privileges. This security model pivots around the concept or a Role. Users (called principals in Tachyon) can be assigned to a Role and it's through a Role they gain permissions to perform actions. Each element of Tachyon's security system leads back to a Role.

2.5.9.7 RBAC objects in Tachyon

Tachyon has several types of objects that are part of its RBAC system. This section serves as a brief description of these objects and their purpose.

2.5.9.7.1 Principals

Principals are identical to users. They're not called users because the word 'user' is traditionally associated with a person. A principal is, from a technical standpoint, an Active Directory account which may be a user account or a computer account. It can also be an AD group.

Tachyon allows access only to principals authenticated through Windows authentication and known to Tachyon itself. Depending on the request, Tachyon establishes the permissions of the calling principal by looking at the roles that principal is assigned to. By default, a fresh installation of Tachyon has a principal representing the user account installing it and another principal for the "NT AUTHORITY\Network Service", which many of Tachyon's services will be running as.

2.5.9.7.2 Roles

All permissions lead to roles. Principals must have roles assigned to them before using the system.

2.5.9.7.3 Permissions

A permission is an ability to perform an operation on a securable type.

2.5.9.7.4 Securable Types

A Securable Type represents a type of an object that can have permissions assigned to it. For example, Instructions can have permissions, as can Consumers and Management Groups. Security itself is an object principals need permissions to, so they can modify it, it will also have a Securable Type. If an element of Tachyon has security defined for it, it must have a securable type.

Instances

An Instance is one, specifc copy of an object of a given Securable Type. This can, for instance, be a single Instruction Set.

While most permissions work on Securable Types as a general thing, where if a Principal is assigned a Permission on a given Securable Type they can work with any object of that type, some Securable Types allow Permissions to be specified on just one given instance of a Securable Type object, giving a more granular control over access to that Securable Type.

As of version 5.0 only InstructionSet securable type supports instances.

2.5.9.7.5 Operations

An Operation (also called Applicable Operation) represents the type of an action that can be performed on a securable type, like "Read" or "Write".

2.5.9.7.6 Example how to read permissions

Marc (**Principal**) through his Global Questioners (**Role**) has Questioner (**Applicable Operation**) permission on Instructions (**Securable type**).

A Principal will have a specific permission on given securable type through a role the principal belongs to. This is the only way principals can obtain permissions and permissions are always for a specific operation on a given type.

To establish the full permission set of a given principal you have to combine all permissions from all the roles assigned to that principal.

2.5.9.8 Retrieving RBAC objects

In this section we will look at endpoints that allow you to examine existing RBAC objects.

2.5.9.8.1 Getting Principals

The most basic functionality is to retrieve all Principals:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Principals will yield following response: **Return payload** Expand source [{ "Id": 1, "ExternalId": "S-1-5-21-1202660629-789336158-1349024091-27850", "PrincipalName": "SomeDomain\\Administrator", "Email": "Administrator@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": true, "DisplayName": "Administrator", "IsGroup": false }, { "Id": 2, "ExternalId": "S-1-5-20", "PrincipalName": "NT AUTHORITY\\Network Service", "Email": null, "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:56.687Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.687Z", "SystemPrincipal": true, "DisplayName": "Network Service", "IsGroup": false }, { "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true,	Use Principals object inside the Tachyon connector instance. **Retrieving all principals** Expand source `principals = onnector.Principals.GetAll();` "principals" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
        "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
        "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
        "SystemPrincipal": false,
        "DisplayName": "Jane Doe",
        "IsGroup": false
    },
    {
        "Id": 4,
        "ExternalId": "S-1-5-21-3276326578-728
399001-2836074973-1009",
        "PrincipalName": "SomeDomain\\John.Doe
",
        "Email": "John.Doe@SomeDomain",
        "Enabled": true,
        "CreatedTimestampUtc": "2016-11-30T15:
10:10.73Z",
        "ModifiedTimestampUtc": "2016-11-30T1
5:10:10.73Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
]
```

Or just a single Principal by their Id. Here we'll look for "SomeDomain\\Jane.Doe", who in has the Id of 3:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Principals/3 will yield following response:
 	
Return payload Expand source
 	
`{`
`  "Id": 3,`
`  "ExternalId": "S-1-5-21-1202660629-7893360 58-1343024091-23842",`
`  "PrincipalName": "SomeDomain\\Jane.Doe",`
`  "Email": "Jane.Doe@SomeDomain.com",`
`  "Enabled": true,`
`  "CreatedTimestampUtc": "2019-11-07T13:14:5 2.777Z",`
`  "ModifiedTimestampUtc": "2019-11-07T13:14: 52.777Z",`
	Use Principals object inside the Tachyon connector instance.
	
	Retrieving specific Principal Expand source
	
	`principal = connector.Princip als.Get(3);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "SystemPrincipal": false,
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```

You can also retrieve all Principals that have a specific Role assigned to them. Here we'll get all Principals who are "Global Administrators", which has the Id of 1:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Principals/Role/1 will yield following response:
 	
Return payload Expand source
 	
[
{
"PrincipalId": 1,
"RoleId": 1,
"CreatedTimestampUtc": "2019-11-07T13:15:01.533Z",
"Role": null,
"Principal": {
"Id": 1,
"ExternalId": "S-1-5-21-1202660629-789336158-1349024091-27850",
"PrincipalName": "SomeDomain\\Administrator",
"Email": "Administrator@SomeDomain.com",
"Enabled": true,
"CreatedTimestampUtc": "2019-11-07T13:14:52.777Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z",
"SystemPrincipal": true,
"DisplayName": "Administrator",
"IsGroup": false
}
}
]
	Use Principals object inside the Tachyon connector instance.
	
	Retrieving all principals that have a specific role Expand source
	
	`principals = connector.Principals.GetForRole(1);`
	
	"principals" object will contain the same data you can see in the JSON response on the left.

2.5.9.8.2 Getting Roles

Much like with Principals, you can retrieve all Roles in the system:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles will yield following response: **Return payload** Expand source [{ "Id": 11, "Name": "1E Client Deployment Administrators", "Description": "This role can create, view and cancel 1E Client deployment jobs", "CreatedTimestampUtc": "2019-11-07T13:14:56.28Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.28Z", "SystemRole": true }, { "Id": 12, "Name": "1E Client Installer Administrators", "Description": "This role can upload, delete and view 1E Client installers", "CreatedTimestampUtc": "2019-11-07T13:14:56.307Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.307Z", "SystemRole": true }, { "Id": 23, "Name": "Applications Administrators", "Description": "This role can upload and delete Applications", "CreatedTimestampUtc": "2019-11-07T13:14:56.693Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.693Z", "SystemRole": true }, { "Id": 17, "Name": "Component Viewers", "Description": "This role can view components", "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",	Use Roles object inside the Tachyon connector instance. **Retrieving all Roles** Expand source `roles = connector.Roles.GetAll();` "roles" object will contain the same data you can see in the JSON response on the left.

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 14,
        "Name": "Connector Administrators",
        "Description": "This role can create, update, d
elete and view connectors; and test them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 10,
        "Name": "Consumer Administrators",
        "Description": "This role can manage Consumers
that use the Tachyon platform",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.523
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.52
3Z",
        "SystemRole": true
    },
    {
        "Id": 9,
        "Name": "Custom Properties Administrators",
        "Description": "This role can add, edit or dele
te custom properties",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.133
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.13
3Z",
        "SystemRole": true
    },
    {
        "Id": 28,
        "Name": "Event Subscription Administrators",
        "Description": "This role has full control over
the Event Subscription configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.90
7Z",
        "SystemRole": true
```

```
    },
    {
        "Id": 29,
        "Name": "Event Subscription Viewers",
        "Description": "This role can view the Event Su
bscription configuration and reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.90
7Z",
        "SystemRole": true
    },
    {
        "Id": 7,
        "Name": "Global Actioners",
        "Description": "This role can ask questions, vi
ew responses and send actions for all instruction sets
",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 1,
        "Name": "Global Administrators",
        "Description": "Has the combined rights of all
the other system roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:52.75
7Z",
        "SystemRole": true
    },
    {
        "Id": 5,
        "Name": "Global Approvers",
        "Description": "This role can approve actions f
or all instruction sets, for anyone other than self",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
```

```
        "Id": 6,
        "Name": "Global Questioners",
        "Description": "This role can ask questions and
view responses for all instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 8,
        "Name": "Global Viewers",
        "Description": "This role can view instructions
and responses for all instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12
Z",
        "SystemRole": true
    },
    {
        "Id": 24,
        "Name": "Guaranteed State Administrators",
        "Description": "This role has full control over
the Guaranteed State configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.71
7Z",
        "SystemRole": true
    },
    {
        "Id": 25,
        "Name": "Guaranteed State Viewers",
        "Description": "This role can view the Guarante
ed State configuration and reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.71
7Z",
        "SystemRole": true
    },
    {
        "Id": 4,
        "Name": "Infrastructure Administrators",
        "Description": "This role can view system statu
```

```
s instrumentation and system information",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.49
3Z",
        "SystemRole": true
    },
    {
        "Id": 2,
        "Name": "Instruction Set Administrators",
        "Description": "This role can add product pack
s; add, modify and delete instruction sets; and delete
instruction definitions",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
        "Id": 19,
        "Name": "Inventory Administrators",
        "Description": "This role can create, update, d
elete and view inventory repositories; populate and arc
hive them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 20,
        "Name": "Inventory Viewers",
        "Description": "This role can view inventory re
positories",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 16,
        "Name": "Log Viewers",
        "Description": "This role can view process, syn
chronization and infrastructure logs",
```

```
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 13,
        "Name": "Management Group Administrators",
        "Description": "This role can create, delete an
d update management groups; and initiate synchronizatio
n of management groups",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.4Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.4Z
",
        "SystemRole": true
    },
    {
        "Id": 21,
        "Name": "Management Group Sync Initiators",
        "Description": "This role can view management g
roups and initiate synchronization of management groups
",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 30,
        "Name": "MySet Viewers",
        "Description": null,
        "CreatedTimestampUtc": "2020-01-02T12:02:12.01Z
",
        "ModifiedTimestampUtc": "2020-01-02T12:02:12.01
Z",
        "SystemRole": false
    },
    {
        "Id": 26,
        "Name": "Patch Success Viewers",
        "Description": "This role can view the Patch Su
ccess dashboards",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.73Z
",
```

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.73
Z",
        "SystemRole": true
    },
    {
        "Id": 3,
        "Name": "Permissions Administrators",
        "Description": "This role can add or remove use
rs; view all roles; add, modify  and delete custom role
s; assign roles to any instruction sets and define thei
r permissions; and view the admin log",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
7Z",
        "SystemRole": true
    },
    {
        "Id": 22,
        "Name": "Permissions Readers",
        "Description": "This role can view all roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 18,
        "Name": "Provider Configuration Administrators
",
        "Description": "This role can update, delete an
d view provider configurations",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 15,
        "Name": "Schedule Administrators",
        "Description": "This role can create, update, d
elete and view schedules; and can view schedule history
",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z
",
```

Direct Consumer API call	C# code using Consumer SDK library

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67
Z",
        "SystemRole": true
    },
    {
        "Id": 27,
        "Name": "ServiceNow ITSM Connect",
        "Description": "This role is used with the Serv
iceNow 1E ITSM Connect application. Once the 1E ITSM Co
nnect application is installed in ServiceNow this role
will be used to grant permissions to the specified Serv
ice Account user.",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.743
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.74
3Z",
        "SystemRole": false
    }
]
```

But unlike the Principals, roles also have an endpoint that supports filtering, sorting and paging. In the example below we'll look for custom roles and sort them by name.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Search with following payload:	Use Roles object inside the Tachyon connector instance.
Request payload Expand source	**Searching for Roles** Expand source

```
{
 "PageSize": 10,
 "Start": 1,
 "Filter": {
 "Attribute": "SystemRole",
 "Operator": "==",
 "Value": "false"
 },
 "Sort": [{
 "Direction": "ASC",
 "Column": "Name"
 }]
}
```

```
var payload = new Se
arch
{
        PageSize = 1
0,
        Start = 1,
        Filter = new
ExpressionObject
        {
                Attr
ibute = "SystemRole
",
                Oper
```

Direct Consumer API call	C# code using Consumer SDK library

Yields following response:
 	
Return payload Expand source
 	
{
"TotalCount": 2,
"Items": [
{
"SystemRole": false,
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 1,
"Id": 30,
"Name": "MySet Viewers",
"Description": null,
"CreatedTimestampUtc": "2020-01-02T12:02:12.01Z",
"ModifiedTimestampUtc": "2020-01-02T12:02:12.01Z"
},
{
"SystemRole": false,
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 0,
"Id": 27,
"Name": "ServiceNow ITSM Connect",
"Description": "This role is used with the ServiceNow 1E ITSM Connect application. Once the 1E ITSM Connect application is installed in ServiceNow this role will be used to grant permissions to the specified Service Account user.",
"CreatedTimestampUtc": "2019-11-07T13:14:56.743Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.743Z"
}
]
}
	ator = "==",
	Value = "false"
	},
	Sort = new List<SortSpec>() { new SortSpec
	{
	Direction = "ASC",
	Column = "Name"
	}}
	};
	
	searchResults = connector.Roles.FindRoles(payload);
	
	"searchResults" object will contain the same data you can see in the JSON response on the left.

You can also retrieve a singel Role by its Id, here we'll retrieve the Global Administrators role which has the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles/1 will yield following response: **Return payload** Expand source `{ "Id": 1, "Name": "Global Administrators", "Description": "Has the combined rights of all the other system roles", "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.757Z", "SystemRole": true }`	Use Roles object inside the Tachyon connector instance. **Retrieving a specifc Role by its Id** Expand source `roles = connector.Roles.Get(1);` "role" object will contain the same data you can see in the JSON response on the left.

Lastly, we'll look at retrieving all Roles given Principal has assigned to them using that Principal's Id. Here we'll get all the Roles that "SomeDomain\Jane.Doe" has assigned and that Principal's Id is 3:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles/Principal/3 will yield following response: **Return payload** Expand source `[{ "PrincipalId": 3, "RoleId": 2, "CreatedTimestampUtc": "2019-11-07T13:14:55.433Z", "Role": { "SystemRole": true, "AssignedManagementGroupCount": 0, "HasAllDevicesManagementGroupAssigned": false, "AssignedPrincipalCount": 1, "Id": 2, "Name": "Instruction Set Administrators", "Description": "This role can add product packs; add, modify and delete instruction sets; and delete instruction definit`	Use Roles object inside the Tachyon connector instance. **Retrieving all Roles fora given Principal** Expand source `roles = connector.Roles.GetForPrincipal(principalId);` "roles" object will contain the same data you can see in the JSON response on the left.

```
ions",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 3,
        "CreatedTimestampUtc": "2019-11-07T1
3:14:54.963Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAss
igned": false,
            "AssignedPrincipalCount": 1,
            "Id": 3,
            "Name": "Permissions Administrat
ors",
            "Description": "This role can ad
d or remove users; view all roles; add, modi
fy  and delete custom roles; assign roles to
any instruction sets and define their permis
sions; and view the admin log",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 5,
        "CreatedTimestampUtc": "2019-11-07T1
3:15:01.533Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAss
igned": false,
            "AssignedPrincipalCount": 1,
            "Id": 5,
```

Direct Consumer API call	C# code using Consumer SDK library


```
            "Name": "Global Approvers",
            "Description": "This role can ap
prove actions for all instruction sets, for
anyone other than self",
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11
-07T13:14:56.117Z"
        },
        "Principal": null
    }
]
```


2.5.9.8.3 Getting Securable Types

You can retrieve all Securable Types with following call:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/SecurableTypes will yield following response: **Return payload** Expand source	Use SecurableTypes object inside the Tachyon connector instance. **Retrieving all Securable Types** Expand source


```
[
    {
        "Id": 1,
        "Name": "InstructionSet",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.113Z",
        "Operations": [
            {
                "Id": 2,
                "OperationName": "Actioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            },
            {
                "Id": 4,
                "OperationName": "Approver",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
```


C# code using Consumer SDK library
`secTypes = connector.Securab leTypes.GetAll();` "secTypes" object will contain the same data you can see in the JSON response on the left.

```
onSet"
            },
            {
                "Id": 3,
                "OperationName": "Questioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            },
            {
                "Id": 1,
                "OperationName": "Viewer",
                "SecurableTypeId": 1,
                "SecurableTypeName": "Instructi
onSet"
            }
        ]
    },
    {
        "Id": 2,
        "Name": "Security",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:52.75Z",
        "Operations": [
            {
                "Id": 7,
                "OperationName": "Delete",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 5,
                "OperationName": "Read",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 6,
                "OperationName": "Write",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            }
        ]
    },
    {
```

```
        "Id": 3,
        "Name": "InstructionSetManagement",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.113Z",
        "Operations": [
            {
                "Id": 9,
                "OperationName": "Add",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            },
            {
                "Id": 8,
                "OperationName": "Delete",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            },
            {
                "Id": 10,
                "OperationName": "Read",
                "SecurableTypeId": 3,
                "SecurableTypeName": "Instructi
onSetManagement"
            }
        ]
    },
    {
        "Id": 4,
        "Name": "Instrumentation",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:52.75Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:52.75Z",
        "Operations": [
            {
                "Id": 11,
                "OperationName": "Read",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Instrumen
tation"
            }
        ]
    },
```

Direct Consumer API call

C# code using Consumer SDK library


```
    {
        "Id": 5,
        "Name": "CustomProperty",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:55.13Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:55.13Z",
        "Operations": [
            {
                "Id": 12,
                "OperationName": "Read",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPro
perty"
            },
            {
                "Id": 13,
                "OperationName": "Write",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomPro
perty"
            }
        ]
    },
    {
        "Id": 6,
        "Name": "Consumer",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:55.523Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:55.523Z",
        "Operations": [
            {
                "Id": 14,
                "OperationName": "Read",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            },
            {
                "Id": 15,
                "OperationName": "Write",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            }
        ]
    },
    {
```

```
        "Id": 7,
        "Name": "AgentDeployment",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.277Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.277Z",
        "Operations": [
            {
                "Id": 18,
                "OperationName": "Approve",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            },
            {
                "Id": 16,
                "OperationName": "Execute",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            },
            {
                "Id": 17,
                "OperationName": "View",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDepl
oyment"
            }
        ]
    },
    {
        "Id": 8,
        "Name": "AgentInstallerManagement",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.307Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.307Z",
        "Operations": [
            {
                "Id": 19,
                "OperationName": "Add",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            },
            {
                "Id": 20,
```

```
                "OperationName": "Delete",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            },
            {
                "Id": 21,
                "OperationName": "Read",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInst
allerManagement"
            }
        ]
    },
    {
        "Id": 9,
        "Name": "ManagementGroup",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.397Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.397Z",
        "Operations": [
            {
                "Id": 29,
                "OperationName": "Delete",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 22,
                "OperationName": "Read",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 24,
                "OperationName": "Synchronize",
                "SecurableTypeId": 9,
                "SecurableTypeName": "Managemen
tGroup"
            },
            {
                "Id": 23,
                "OperationName": "Write",
                "SecurableTypeId": 9,
```

```
                "SecurableTypeName": "Managemen
tGroup"
            }
        ]
    },
    {
        "Id": 10,
        "Name": "Connector",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 27,
                "OperationName": "Delete",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 28,
                "OperationName": "Execute",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 25,
                "OperationName": "Read",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            },
            {
                "Id": 26,
                "OperationName": "Write",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector
"
            }
        ]
    },
    {
        "Id": 11,
        "Name": "Schedule",
        "CreatedTimestampUtc": "2019-11-07T13:1
```

```
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 32,
                "OperationName": "Delete",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 30,
                "OperationName": "Read",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 31,
                "OperationName": "Write",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            }
        ]
    },
    {
        "Id": 12,
        "Name": "ProcessLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 33,
                "OperationName": "Read",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLo
g"
            }
        ]
    },
    {
        "Id": 13,
        "Name": "SynchronizationLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
```

```
14:56.663Z",
        "Operations": [
            {
                "Id": 34,
                "OperationName": "Read",
                "SecurableTypeId": 13,
                "SecurableTypeName": "Synchroni
zationLog"
            }
        ]
    },
    {
        "Id": 14,
        "Name": "Component",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 35,
                "OperationName": "Read",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component
"
            }
        ]
    },
    {
        "Id": 15,
        "Name": "ProviderConfiguration",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 38,
                "OperationName": "Delete",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderC
onfiguration"
            },
            {
                "Id": 36,
                "OperationName": "Read",
                "SecurableTypeId": 15,
```

```
                "SecurableTypeName": "ProviderC
onfiguration"
            },
            {
                "Id": 37,
                "OperationName": "Write",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderC
onfiguration"
            }
        ]
    },
    {
        "Id": 16,
        "Name": "InfrastructureLog",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 39,
                "OperationName": "Read",
                "SecurableTypeId": 16,
                "SecurableTypeName": "Infrastru
ctureLog"
            }
        ]
    },
    {
        "Id": 17,
        "Name": "Repository.Inventory",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.663Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.663Z",
        "Operations": [
            {
                "Id": 43,
                "OperationName": "Archive",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 42,
                "OperationName": "Delete",
```

```
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 53,
                "OperationName": "EvaluateManag
ementGroups",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 44,
                "OperationName": "Populate",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 40,
                "OperationName": "Read",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            },
            {
                "Id": 41,
                "OperationName": "Write",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repositor
y.Inventory"
            }
        ]
    },
    {
        "Id": 18,
        "Name": "Application",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.693Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.693Z",
        "Operations": [
            {
                "Id": 46,
                "OperationName": "Delete",
                "SecurableTypeId": 18,
```

```
                "SecurableTypeName": "Applicati
on"
            },
            {
                "Id": 45,
                "OperationName": "Write",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Applicati
on"
            }
        ]
    },
    {
        "Id": 19,
        "Name": "GuaranteedState",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.713Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.713Z",
        "Operations": [
            {
                "Id": 49,
                "OperationName": "Delete",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            },
            {
                "Id": 47,
                "OperationName": "Read",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            },
            {
                "Id": 48,
                "OperationName": "Write",
                "SecurableTypeId": 19,
                "SecurableTypeName": "Guarantee
dState"
            }
        ]
    },
    {
        "Id": 20,
        "Name": "Repository.Patch",
        "CreatedTimestampUtc": "2019-11-07T13:1
```

```
4:56.73Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.73Z",
        "Operations": [
            {
                "Id": 50,
                "OperationName": "Read",
                "SecurableTypeId": 20,
                "SecurableTypeName": "Repositor
y.Patch"
            }
        ]
    },
    {
        "Id": 21,
        "Name": "Repository.BI",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.737Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.737Z",
        "Operations": [
            {
                "Id": 52,
                "OperationName": "Populate",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repositor
y.BI"
            },
            {
                "Id": 51,
                "OperationName": "Read",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repositor
y.BI"
            }
        ]
    },
    {
        "Id": 22,
        "Name": "EventSubscription",
        "CreatedTimestampUtc": "2019-11-07T13:1
4:56.907Z",
        "ModifiedTimestampUtc": "2019-11-07T13:
14:56.907Z",
        "Operations": [
            {
                "Id": 56,
```

```
                "OperationName": "Delete",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            },
            {
                "Id": 54,
                "OperationName": "Read",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            },
            {
                "Id": 55,
                "OperationName": "Write",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubs
cription"
            }
        ]
    },
    {
        "Id": 24,
        "Name": "Repository.AppMigration",
        "CreatedTimestampUtc": "2019-11-26T09:3
4:40.923Z",
        "ModifiedTimestampUtc": "2019-11-26T09:
34:40.923Z",
        "Operations": [
            {
                "Id": 61,
                "OperationName": "Archive",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 60,
                "OperationName": "Execute",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 58,
                "OperationName": "Read",
                "SecurableTypeId": 24,
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 62,
                "OperationName": "Whatever",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            },
            {
                "Id": 59,
                "OperationName": "Write",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repositor
y.AppMigration"
            }
        ]
    }
]
```

You can also retrieve a specific Securable Type by either its Name or Id.

Direct Consumer API call	C# code using Consumer SDK library
If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/1 If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/Name/InstructionSet Both yield following response: **Return payload** Expand source	Use SecurableTypes object inside the Tachyon connector instance. To retrieve a Securable Type using its Id use: **Retrieving a Securable Type by Id** Expand source `secType = connector.SecurableTypes.Get(1);` To retrieve a Securable Type using its Name use: **Retrieving Securable Type by Name** Expand source

```
{
    "Id": 1,
    "Name": "InstructionSet",
    "CreatedTimestampUtc": "2019-11-07T13:14:52.75Z
",
    "ModifiedTimestampUtc": "2019-11-07T13:14:56.11
3Z",
    "Operations": [
        {
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"Id": 2,
"OperationName": "Actioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 4,
"OperationName": "Approver",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 3,
"OperationName": "Questioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 1,
"OperationName": "Viewer",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
}
]
}</pre>
	<pre>secType = connector.Secu rableTypes.Get("Instruct ionSet");</pre>
	"secType" object will contain the same data you can see in the JSON response on the left.

2.5.9.8.4 Getting Applicable Operations

Applicable Operations work in the context of a Securable Type, which is why Tachyon allows you to retrieve all Applicable Operations for given Securable Type. You can use either Name or Id to specify which Securable Type you want Applicable Operations for.

Direct Consumer API call	C# code using Consumer SDK library
If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableTypeId/1 If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/ApplicableOperations/SecurableType Name/InstructionSet Both yield the following response:	Use ApplicableOperations object inside the Tachyon connector instance. To retrieve Applicable Operations for a Securable Type using the type's Id use: **Retrieve Applicable Operations using Securable Type's Id** Expand source <pre>operations = connector.Applicable Operations.Get(1);</pre>

Direct Consumer API call	C# code using Consumer SDK library

Return payload Expand source
 	
[
{
"Id": 2,
"OperationName": "Actioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 4,
"OperationName": "Approver",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 3,
"OperationName": "Questioner",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
},
{
"Id": 1,
"OperationName": "Viewer",
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet"
}
]
	To retrieve Applicable Operations for a Securable Type using the type's Name use:
	
	Retrieve Applicable Operations using Securable Type's Name Expand source
	
	operations = connector.ApplicableOperations.Get("InstructionSet");
	
	"operations" object will contain the same data you can see in the JSON response on the left.

2.5.9.8.5 Getting Permissions

You can retrieve permissions in several different ways, for example for a Principal, a Role or a Securable Type.

Some APIs perform checks for the calling user by pulling the user information from the HTTP request itself. Other APIs allow you to specify which object you want to get permissions for.

2.5.9.8.5.1 Getting permissions for a Principal

Getting all Permissions

Retrieving a principal's permissions is done using the account name (for example "somedomain\jane.doe") of that principal. When directly using the API, you have to encode that account name into base64 before sending it. C# Consumer API SDK will do the encoding for you.

In general, any GET endpoint requires you to base64 encode the principal name, due to the fact that principal names can contain characters that are not allowed in URIs.

When you request permissions for a specific Principal, you'll get the permissions that Principal has stemming from any of their roles.

The following examples use the "somedomain\jane.doe" account.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ== will yield following response: **Return payload** Expand source `[{ "SecurableId": null, "SecurableName": null, "SecurableTypeId": 4, "SecurableTypeName": "Instrumentation", "RoleId": 4, "RoleName": "Infrastructure Administrators", "Allowed": true, "Operations": [{ "PermissionId": 19, "OperationId": 11, "OperationName": "Read", "CreatedTimestampUtc": "2019-06-07T15:12:40.757Z", "ModifiedTimestampUtc": "2019-06-07T15:12:40.757Z" }] }, { "SecurableId": null, "SecurableName": null, "SecurableTypeId": 12, "SecurableTypeName": "ProcessLog", "RoleId": 16, "RoleName": "Log Viewers", "Allowed": true, "Operations": [{ "PermissionId": 61, "OperationId": 33, "OperationName": "Read", "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z", "ModifiedTimestampUtc": "2019-06-07T15:12:`	Use Permissions object inside the Tachyon connector instance. **Retrieving all Permissions for a specific Principal** Expand source `permissions = connector.Permissions.GetForPrincipal("somedomain\\jane.doe");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 13,
    "SecurableTypeName": "SynchronizationLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 62,
        "OperationId": 34,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 14,
    "SecurableTypeName": "Component",
    "RoleId": 17,
    "RoleName": "Component Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 64,
        "OperationId": 35,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 16,
    "SecurableTypeName": "InfrastructureLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
```

Direct Consumer API call	C# code using Consumer SDK library

```
    "Operations": [{
        "PermissionId": 63,
        "OperationId": 39,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:4
4.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:
44.13Z"
    }]
}]
```

Checking for a specific Permission

You can also retrieve permissions given a Principal has on a particular Securable Type.

In the example below, we'll look at InstructionSet related permissions "somedomain\jane.doe" account has.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqYW5lLmRvZQ==/Type/InstructionSet will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving InstructionSet related permissions for a Principal** Expand source `permissions = connector.Permissions.GetForPrincipalAndType("somedomain\\jane.doe", "InstructionSet");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1,
        "RoleName": "Global Administrators",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
```

```
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07
T13:14:52.77Z",
                "ModifiedTimestampUtc": "2019-11-0
7T13:14:52.77Z"
            }
        ]
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

If given Securable Type supports Instances, you can also check a Principal's permissions on a specific instance.

In the example below we'll use a different account called "somedomain\john.doe" and check what permissions it has on Instruction Set with the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/Type/InstructionSet/1 will yield following response: **Return payload** Expand source `[{ "SecurableId": 1, "SecurableName": "MySet", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTimestampUtc": "2020-01-02T12:04:04.963Z" }] }]`	Use Permissions object inside the Tachyon connector instance. **Retrieving Principal's permissions to a specific instruction set** Expand source `permissions = connector.Permissions.GetForPrincipalAndTypeAndInstance("somedomain\\john.doe", "InstructionSet", 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

If the given Principal has no permissions or does not have permissions on given Securable Type and/or instance, an empty collection is returned.

2.5.9.8.5.2 Getting permissions for a Role

You can retieve Role's permissions by using the role Id.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16 will yield following response:
 	
Return payload Expand source
 	
[
{
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 12,
"SecurableTypeName": "ProcessLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [
{
"PermissionId": 61,
"OperationId": 33,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-11-07T13:14:56.673Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.673Z"
}
]
},
{
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 13,
"SecurableTypeName": "SynchronizationLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [
{
"PermissionId": 62,
"OperationId": 34,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-11-07T13:14:56.673Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.673Z"
}
]
	Use Permissions object inside the Tachyon connector instance.
	
	Retrieving all permissions assigned to a Role Expand source
	
	`permissions = connector.Permissions.GetForRole(16);`
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 16,
        "SecurableTypeName": "Infrastructur
eLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 63,
                "OperationId": 39,
                "OperationName": "Read",
                "CreatedTimestampUtc": "201
9-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc": "20
19-11-07T13:14:56.673Z"
            }
        ]
    }
]
```

You can also retrieve all permissions given Role has on a specific Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16/Type/ProcessLog will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions that a Role has on a given Securable Type** Expand source `permissions = connector.Permissions.GetForRoleAndType(16, "ProcessLog");` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 12,
        "SecurableTypeName": "ProcessLog
",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
```

Direct Consumer API call	C# code using Consumer SDK library

```
                "PermissionId": 61,
                "OperationId": 33,
                "OperationName": "Read",
                "CreatedTimestampUtc": "
2019-11-07T13:14:56.673Z",
                "ModifiedTimestampUtc":
"2019-11-07T13:14:56.673Z"
            }
        ]
    }
]
```

And for Securable Types that support Instances, you can also check Permissions on a specific Instance.

Here we'll get permissions for a custom Role that's been assigned a 'Viewer' permissions on an Instruction Set.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/30/Type/InstructionSet/1 will yield following response: **Return payload** Expand source	Use Permissions object inside the Tachyon connector instance. **Retrieving permissions a Role has on a specific Instance of a Securable Type** Expand source `permissions = connector.Permissions.GetForRoleAndTypeAndInstance(30, "InstructionSet", 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "SecurableId": 1,
        "SecurableName": "Wszystko",
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instructi
onSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewe
r",
                "CreatedTimestampUtc":
"2020-01-02T12:04:04.963Z",
                "ModifiedTimestampUtc":
"2020-01-02T12:04:04.963Z"
            }
        ]
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

2.5.9.8.5.3 Getting permissions for a Securable Type

You can retrieve permissions granted on a Securable Type using the type's Id. This will return a collection of all permissions on a given type assigned to any of the Roles.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Securable/1 will yield following response:

Return payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 1,
        "RoleName": "Global Administrator
s",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
```

Use Permissions object inside the Tachyon connector instance.

Retrieving all Permisisons on a given Securable Type Expand source

```
permissions = connector.Permission
s.GetForSecurableType(1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approve
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:52.77Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
```

```
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 6,
        "RoleName": "Global Questioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 21,
                "OperationId": 3,
                "OperationName": "Questio
ner",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 7,
        "RoleName": "Global Actioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 22,
                "OperationId": 2,
                "OperationName": "Actione
r",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 8,
```

Direct Consumer API call	C# code using Consumer SDK library


```
        "RoleName": "Global Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 23,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:14:54.96Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "Instruction
Set",
        "RoleId": 27,
        "RoleName": "ServiceNow ITSM Conn
ect",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 136,
                "OperationId": 1,
                "OperationName": "Viewer
",
                "CreatedTimestampUtc": "2
019-11-07T13:18:05.767Z",
                "ModifiedTimestampUtc": "
2019-11-07T13:18:05.767Z"
            }
        ]
    }
]
```


You can also retrieve permissions granted on a specific Instance of a Securable Type.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Perm	Use Permissions object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
issions/Securable/1/1 will yield following response: **Return payload** Expand source `[{ "SecurableId": 1, "SecurableName": null, "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet", "RoleId": 30, "RoleName": "MySet Viewers", "Allowed": true, "Operations": [{ "PermissionId": 137, "OperationId": 1, "OperationName": "Viewer", "CreatedTimestampUtc": "2020-01-02T12:04:04.963Z", "ModifiedTimestampUtc": "2020-01-02T12:04:04.963Z" }] }]`	**Retrieving permissions assigned on a specific instance of a Securable Type** Expand source `connector.Permissions.GetForSecurableTypeAndInstance(1, 1);` "permissions" object will contain the same data you can see in the JSON response on the left.

2.5.9.8.5.4 Helper APIs

Consumer API has a number of endpoints that are not directly involved with its RBAC system but instead provide various utility functions around the general principal and permission area.

Who am I?

You can ask the API to return the information about the user who's making the request. The API examines the incoming request, retrieves the name of the user from it, makes sure the user is authenticated, and returns information summary about the user, like user's principal name, SID, display name and email.

This endpoint is useful for systems that cannot assume they're running under a particular account, like a browser or an application running in the context of launching user. In those cases this endpoint can be used to obtain the name of the user, which is then fed to the Permissions endpoint seen above in order to retrieve the permissions of the caller.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/whoami will yield following response: **Return payload** Expand source `{ "PrincipalName": "Somedomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-19970", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "Photo": null }`	Use PrincipalSearch object inside the Tachyon connector instance. **Who am I?** Expand source `user = connector.PrincipalSearch.GetCurrentlyLoggedInUser();` "user" object will contain the same data you can see in the JSON response on the left.

Active Directory search

Because in order to add a Principal to Tachyon you have to know the SID of the user, Tachyon's Consumer API exposes endpoints that help you look up users in Active Directory and retrieve information about them, including their SID.

You can perform a search for Active Directory accounts whose common name or sAMAccountName contain given phrase. Details of what exact search is performed are beyond the scope of this page and depend on type of objects being searched for, like users, groups and computers and whether your Tachyon installation is using GC or LDAP.

In the examples below we'll be looking for accounts that contain the phrase "Administrator". How you perform this search depends on your version of Tachyon.

Tachyon versions up to and including 4.1

You simply provide a base64 encoded search string and issue a GET request.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/QWRtaW5pc3RyYXRvcg== will yield following response: **Return payload** Expand source `[ { "PrincipalName": "SomeDomain\\CSU`	Use PrincipalSearch object inside the Tachyon connector instance. **Searching for principals in Active Directory** Expand source

Direct Consumer API call	C# code using Consumer SDK library

<pre>serAdministrator",
"ExternalId": "S-1-5-21-120266062
9-789336058-1353024091-8145",
"Email": null,
"DisplayName": "CSUserAdministrat
or",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\CSV
oiceAdministrator",
"ExternalId": "S-1-5-21-120266062
9-789336058-1343023091-8024",
"Email": null,
"DisplayName": "CSVoiceAdministra
tor",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exc
hange Organization Administrators",
"ExternalId": "S-1-5-21-120266062
9-789336058-1343324091-6881",
"Email": null,
"DisplayName": "Exchange Organiza
tion Administrators",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exc
hange Public Folder Administrators",
"ExternalId": "S-1-5-21-120266062
9-789336058-1340024091-7747",
"Email": null,
"DisplayName": "Exchange Public F
older Administrators",
"IsGroup": true
},
{
"PrincipalName": "SomeDomain\\Exc
hange Recipient Administrators",
"ExternalId": "S-1-5-21-120266062
9-789336058-1340024091-6913",
"Email": null,
"DisplayName": "Exchange Recipien
t Administrators",
"IsGroup": true</pre>
	<pre>results = connector.Princi
	palSearch.SearchForPrincip
	als("Administrator");</pre>
	"results" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
        },
        {
                "PrincipalName": "SomeDomain\\Exc
hange View-Only Administrators",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange View-Onl
y Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Sec
urity Administrator",
                "ExternalId": "S-1-5-21-120266062
9-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security Administ
rator",
                "IsGroup": true
        }
]
```

Tachyon versions 5.0 and newer

Here you have greater control over the search. A POST request should be used and in its payload you can specify, apart from the search string, number of results returned, sort column and order.

You can search for "user" and "group" object types and sort in either ascending (using "ASC") or descending order (using "DESC") on following columns: "cn", "mail", "sAMAccountName", "description", "objectSid", "displayName". Column names, object types and sorting direction string are not case sensitive.

You have to provide the SearchText (the text to search for) and at least one object type. Pagesize and sort are optional and if not provided will default to respectively 100 and ascending sorting on display name.

One things that you have to keep in mind when it comes to PageSize property is that, due to internal implementation, it defines the size of the page returned by Active Directory. This means that you can get fewer results than you requested, even if there would have been more to return because Tachyon filters out users already in the system. As an example, if you define the page size to be 10 and the Active Directory has 18 entries that match your search string, it will return 10 entries. Now let us assume that out of those 10 entries 1 user is already in Tachyon. This user will be filtered out of the return data set, so you will receive only 9 entries.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalSearch with following payload:	Use PrincipalSearch object inside the Tachyon connector instance.

Direct Consumer API call

Request payload Expand source

```
{
        "SearchText": "Administrator",
        "ObjectTypes": ["user", "group
"],
        "PageSize": 100,
        "Sort": {
                "Column": "displayname",
                "Direction": "ASC"
        }
}
```

will yield following response:

Return payload Expand source

```
[
        {
                "PrincipalName": "SomeDo
main\\CSUserAdministrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1353024091-8145",
                "Email": null,
                "DisplayName": "CSUserAd
ministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\CSVoiceAdministrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343023091-8024",
                "Email": null,
                "DisplayName": "CSVoiceA
dministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Organization Administrato
rs",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343324091-6881",
                "Email": null,
                "DisplayName": "Exchange
Organization Administrators",
                "IsGroup": true
```

C# code using Consumer SDK library

Retrieving 100 entries sorted by displayna me Expand source

```
var searchParams = new ActiveDirect
orySearchModel
{
        SearchText = "Administrator
",
        ObjectTypes = new List<stri
ng> { "user", "group" },
        PageSize = 100,
        Sort = new SortSpec
        {
                Column = "displayna
me",
                Direction = "ASC"
        }
};

results = connector.PrincipalSearc
h.SearchForPrincipals(searchParam
s);
```

"results" object will contain the same data you can see in the JSON response on the left.

```
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Public Folder Administrat
ors",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-7747",
                "Email": null,
                "DisplayName": "Exchange
Public Folder Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange Recipient Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange
Recipient Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Exchange View-Only Administrators
",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange
View-Only Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDo
main\\Security Administrator",
                "ExternalId": "S-1-5-21-
1202660629-789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security
Administrator",
                "IsGroup": true
        }
]
```

2.5.9.8.5.5 Retrieving Active Directory information about a specific account

You can also retrieve user's information by their principal name, though this endpoint only returns authenticated users who are also Principals in Tachyon (either directly or throught group membership).

This means that in order to obtain information about somedomain\jane.doe using this endpoint, somedomain\jane.doe has to be a domain account that's either a Principal in Tachyon, or belongs to an Active Directory group that is a Principal in Tachyon.

When using the API directly, the principal name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/user/c29tZWRvbWFpblxqYW5lLmRvZQ==will yield following response: **Return payload** Expand source `[ { "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "jane.doe@somedomain.com", "DisplayName": "Jane Doe", "IsGroup": false } ]`	Use PrincipalSearch object inside the Tachyon connector instance. **Retrieve information about a principal from Active Directory** Expand source `results = connector.PrincipalSearch.SearchForUser("SomeDomain\\Jane.Doe");` "results" object will contain the same data you can see in the JSON response on the left.

Retrieving members of an Active Directory group

You can ask Tachyon to retrieve members of an Active Directory group. This group does not have to be a principal in Tachyon for this functionality to work.

When using the API directly the group name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

In the example below we'll use an AD group called "Tachyon users" and retrieve its members.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/GetMembers/VGFjaHlvbiB1c2Vycw==will yield following response:	Use PrincipalSearch object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

Return payload Expand source
 	
[
{
"PrincipalName": "SomeDomain\\Jane.Doe",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842",
"Email": "jane.doe@somedomain.com",
"DisplayName": "Jane Doe",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Meetra.Surnik",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-8631",
"Email": "Meetra.Surnik@somedomain.com",
"DisplayName": "Meetra Surnik",
"IsGroup": false
},
{
"PrincipalName": "SomeDomain\\Keiran.Halcyon",
"ExternalId": "S-1-5-21-1202660629-789336058-1343024091-1138",
"Email": "Keiran.Halcyon@somedomain.com",
"DisplayName": "Keiran Halcyon",
"IsGroup": false
}
]
	Retrieving members of an Aciv e Directory group Expand source
	
	`results = connector.PrincipalSearch.GetGroupMembers("Tachyon users");`
	
	"results" object will contain the same data you can see in the JSON response on the left.

2.5.9.9 Configuring Tachyon RBAC through the Consumer API

In this section we will look at configuring RBAC emphasizing operations on Principals, Roles and Permissions but also covering Securable Types and Operations later on.

2.5.9.9.1

2.5.9.9.1.1 Adding Principals from Active Directory

Adding Principals to Tachyon will be one of the first things you'll do after installing the system. A fresh installation of Tachyon (which in this context means an installation where there was no previous Master database or such database was dropped) will have two Principals - the account used to install the Tachyon Server (and created the database to be specific) and NT AUTHORITY\Network Service account.

These accounts have limited permissions so to properly use the system you should add more Principals. In order to add a principal to Tachyon you will need an External Id, also called SID, for an active directory account that you wish to add as a principal. Although you can obtain account details through various means. In Active Directory search we've seen Tachyon exposes endpoints that return this information and enables you to search for Active Directory accounts.

Here we'll assume you already have the account details and focus on adding an account as a Principal to Tachyon.

In the example below we'll "SomeDomain\Jane.Doe" account and add it to Tachyon.

To add an account as a Principal you'll at the very least need to supply an ExternalId (SID) and Principal Name, though its advised to also supply a Display Name. Also, unless you set the Enabled flag to true, the newly created account will be disabled by default.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Principals with following payload:
 	
Request payload Expand source
 	
`{ "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "Jane.Doe@SomeDomain.com", "DisplayName": "Jane Doe", "IsGroup": false, "Enabled": true }`
 	
Will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 3, "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": "Jane.Doe@SomeDomain.com", "Enabled": true, "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z", "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z", "SystemPrincipal": false, "DisplayName": "Jane Doe",`
	Use Principals object inside the Tachyon connector instance.
	
	Adding a Principal to Tachyon Expand source
	
	`var payload = new Principal { PrincipalName = "SomeDomain\\Jane.Doe", ExternalId = "S-1-5-21-1202660629-789336058-1343024091-23842", Email = "Jane.Doe@SomeDomain.com", DisplayName = "Jane Doe", IsGroup = false, Enabled = true };`
	
	`principal = connector.Principals.Add(payload);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    "IsGroup": false
}
```

Note

You cannot add a Principal with the same PrincipalName or ExternalId as another Principal that's already in the system, because both of those values must be unique.

Updating a Principal

Updating a Principal is very similar to adding one. You have to use a PUT verb instead of a POST if using the API directly, or Update method instead of Add if you're using the SDK. You also have to provide an Id that belongs to an existing Principal. Remember, you cannot modify system Principals.

Again, you have to remember that ExternalId and PrincipalName must remain unique so you cannot change Principal's PrincipalName or ExternalId to one that matches another Principal's.

2.5.9.9.2 Configuring Roles

Roles are the pivotal point of an RBAC system.

Tachyon comes with a number of pre-defined roles, which should be sufficient to start with, but in time you will most likely create custom roles, either to restrict access to instruction sets or to cover sets of permission required by specific roles within your organization.

Adding, Editing and Removing a Role

First let's look at creating a Role. This can be done either in isolation, where just the role is created and any permissions must be assigned in subsequent call(s), or as a more complete package, where a Role is created along with a set of Permissions, which can include Management Groups.

Regardless of which approach is used, Principals have to be assigned separately.

To create a Role all you require is a name, and this name must be unique. Description can optionally be provided and you can't create system roles.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roleswith following payload: **Request payload** Expand source `{ "Name": "Custom role", "Description": "this is a description" }` Will yield following response:	Use Roles object inside the Tachyon connector instance. **Creating a Role** Expand source `var payload = new Role { Name = "Custom role", Description = "this is a description" };`

Direct Consumer API call	C# code using Consumer SDK library

Return payload Expand source
 	
{
"Id": 31,
"Name": "Custom role",
"Description": "this is a description",
"CreatedTimestampUtc": "2020-01-21T13:36:55.8956916Z",
"ModifiedTimestampUtc": "2020-01-21T13:36:55.8956916Z",
"SystemRole": false
}
	role = connector.Roles.Add(payload);
	
	"role" object will contain the same data you can see in the JSON response on the left.

Creating a Role with Permissions is possible by joining Role creation we've just seen with creating permissions, which is described a bit later.

This effectively means that we take a regular Role creation payload and add a 'Permissions' property to it. That propery is an array of permissions, which we are yet to look at in greater details.

The example below creates a role, which has "Viewer" and "Approver" permissions on Instruction Set with the Id of 1 and "Actioner", "Questioner", "Viewer" and "Approver" permissions on Instruction Set with the Id of 2. It also has permissions on three Management Groups.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Roles/Completewith following payload:
 	
Request payload Expand source
 	
{
"Name": "Complete Role 1",
"Description": "This is a test role",
"Permissions": [{
"SecurableId": 2,
"SecurableTypeId": 1,
"Allowed": true,
"Operations": [{
"OperationId": 2
},
{
"OperationId": 4
},
{
"OperationId": 3
	Use Roles object inside the Tachyon connector instance.
	
	Creating a Role with Permissions Expand source
	
	var payload = new Tachyon.SDK.Consumer.Models.Send.RoleWithPermissionsAndManagementGroups
	{
	Name = "Complete Role 1",
	Description = "This is a test role",
	Permissions = new AggregatedPermission[]
	{
	new AggregatedPermission
	{

Direct Consumer API call

```
        },
        {
            "OperationId": 1
        }]
    },
    {
        "SecurableId": 1,
        "SecurableTypeId": 1,
        "Allowed": true,
        "Operations": [{
            "OperationId": 1
        },
        {
            "OperationId": 4
        }]
    }],
    "ManagementGroupIds": [10,11,12]
}
```

Will yield following response:

Return payload Expand source

```
{
    "Permissions": [
        {
            "SecurableId": 1,
            "SecurableName": null,
            "SecurableTypeId": 1,
            "SecurableTypeName": "InstructionSet
",
            "RoleId": 32,
            "RoleName": "Complete Role 1",
            "Allowed": true,
            "Operations": [
                {
                    "PermissionId": 142,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 143,
                    "OperationId": 4,
```

C# code using Consumer SDK library

```
            SecurableId =
2,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 2 },
                new Permiss
ionOperation { OperationId
= 3 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        },
        new AggregatedPermi
ssion
        {
            SecurableId =
1,
            Allowed = true,
            SecurableTypeId
= 1,
            Operations = ne
w List<PermissionOperation>
            {
                new Permiss
ionOperation { OperationId
= 1 },
                new Permiss
ionOperation { OperationId
= 4 }
            }
        }
    },
    ManagementGroupIds = ne
w List<int> { 10, 11, 12 }
};
```

Direct Consumer API call	C# code using Consumer SDK library

<pre>"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
}
]
},
{
"SsecurableId": 2,
"SecurableName": null,
"SecurableTypeId": 1,
"SecurableTypeName": "InstructionSet",
"RoleId": 32,
"RoleName": "Complete Role 1",
"Allowed": true,
"Operations": [
{
"PermissionId": 138,
"OperationId": 2,
"OperationName": "Actioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 139,
"OperationId": 4,
"OperationName": "Approver",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{
"PermissionId": 140,
"OperationId": 3,
"OperationName": "Questioner",
"CreatedTimestampUtc": "2020-01-22T07:18:57.73Z",
"ModifiedTimestampUtc": "2020-01-22T07:18:57.73Z"
},
{</pre>
	<pre>role = connector.Roles.Add(payload);</pre>
	"role" object will contain the same data you can see in the JSON response on the left.

```
                    "PermissionId": 141,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc": "2020
-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc": "202
0-01-22T07:18:57.73Z"
                }
            ]
        }
    ],
    "ManagementGroups": [
        {
            "Id": 10,
            "Name": "Management group number 9 f
rom repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 9 from repositor
y Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 1,
            "UsableId": "9",
            "HashOfMembers": "4CFD42DAA10FD0FDE0
B4E6BE1116155AC6180EE4E9ADF543F39E97D9B8C51B25",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.677Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.343Z"
        },
        {
            "Id": 11,
            "Name": "Management group number 10
from repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 10 from reposito
ry Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 10,
            "UsableId": "10",
            "HashOfMembers": "F51DE677220883E4A6
84DBB7D965110DE74517D714B3459DA4A98C3B56C58D1C",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.68Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.43Z"
```

```
        },
        {
            "Id": 12,
            "Name": "Management group number 11
from repository Default Inventory",
            "Description": "This is the descript
ion for Management group number 11 from reposito
ry Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 27,
            "UsableId": "11",
            "HashOfMembers": "5D25CC2DEBF1D76B81
602884623FDE878FE8EF57568A6F97A9F04242FEF2655D",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.68Z",
            "ModifiedTimestampUtc": "2020-01-08T
16:31:03.517Z"
        }
    ],
    "Id": 32,
    "Name": "Complete Role 1",
    "Description": "This is a test role",
    "CreatedTimestampUtc": "2020-01-22T07:18:57.
7082854Z",
    "ModifiedTimestampUtc": "2020-01-22T07:18:5
7.7082854Z",
    "SystemRole": false
}
```


To update a Role you should use PUT versions of the two endpoints seen above and provide one additional property in the JSON payload - the Id of the Role being modified. If you're using the Consumer SDK, you should call Update method instead of Add and you must provide an Id in the payload object sent to the API.

Since the names of Roles have to be unique, you won't be able to change the name of a Role to one that is the same as another Role.

Changing the Role details with https://my.tachyon.server/Consumer/Roles (or using the Update method in Consumer SDK that takes Role object type) will change just the details of the role will not change the permissions in any way. It will not change which Mangement Groups or Principals are assigned to the Role either.

Changing the Role via https://my.tachyon.server/Consumer/Roles/Complete will not change which Principals are assigned to the Role, but it will change the Permissions and Management Groups. When using this endpoint with PUT verb (or using the Update method in Consumer SDK that takes RoleWithPermissionsAndManagementGroups object type), all of the Permissions for given Role will be replaced with Permissions provided in the payload and all Management Groups will be replaced with Management Groups provided in the payload. This means that if you do not provide Permissions (omit the property, provide a

null value or an empty array), all Permissions will be removed from the Role. Likewise, if you do not provide ManagementGroups all Management Groups will be unassigned from the Role.

Deleting a role is straightforward and you just need its Id. You can also delete multiple Roles in one call by providing a collection of Role Ids. In either case, you cannot delete a system Role.

Direct Consumer API call	C# code using Consumer SDK library

To delete a single Role make a DELETE request to https://my.tachyon.server/Consumer/Roles/31 ,which will delete the Role with the Id of 31.
 	
To delete multiple Roles, make a DELETE request and send an array of Ids to https://my.tachyon.server/Consumer/Roles. To delete Roles with Ids 31, 32, 33 and 34 you would send this payload:
 	
Request payload Expand source
 	
`[31,32,33,34]`
	Use Roles object inside the Tachyon connector instance.
	
	To delete a single Role:
	
	Deleting the Role with Id 31 Expand source
	
	`connector.Roles.Delete(31);`
	
	To delete multiple Roles:code
	
	Deleting Roles with Ids 31,32,33 and 34 . Expand source
	
	`connector.Roles.DeleteMultiple(new List<int> {31, 32, 33, 34});`

Assigning and unassigning Principals to Roles

As we've seen above when we covered retrieving Principals and Roles, the association between Principals and Roles can be examined from either end: which Principals are assigned to a given Role or which Roles given Principal has assigned.

Looking at the issue from the perspective of a Principal, you can assign or unassign Roles(s) to a Principal, or replace any Roles that Principal has with a given set of Roles.

First, let's pick a Principal that we will assign Roles to. In the example below I'll use "SomeDomain\\Jane.Doe", which has the Id of 3 and I'll assign 4 Roles to that Principal: Instruction Set Administrators, Permissions Administrators, Infrastructure Administrators, Global Questioners which have respective Ids of 2, 3, 4 and 6.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Request payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source `[2,3,4,6]`	Use PrincipalRoles object inside the Tachyon connector instance. **Adding multiple Roles to a Principal** Expand source `roles = connector.PrincipalRoles.AddRolesToPri`

Direct Consumer API call	C# code using Consumer SDK library

will yield following response:
 	
Return payload Expand source
 	
[
{
"PrincipalId": 3,
"RoleId": 2,
"CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
"Role": {
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 2,
"Id": 2,
"Name": "Instruction Set Administrators",
"Description": "This role can add product packs; add, modify and delete instruction sets; and delete instruction definitions",
"CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
"SystemRole": true
},
"Principal": null
},
{
"PrincipalId": 3,
"RoleId": 3,
"CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
"Role": {
"AssignedManagementGroupCount": 0,
"HasAllDevicesManagementGroupAssigned": false,
"AssignedPrincipalCount": 2,
"Id": 3,
"Name": "Permissions Administrators",
"Description": "This role can add or remove users; view all roles; add, modify and delete custom roles; assign roles to any instruction sets and define their permissions; and view the admin log",
"CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
	ncipal(3, new List<int> { 2, 3, 4, 6 });
	
	"roles" object will contain the same data you can see in the JSON response on the left.

```
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.117Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 4,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.
31Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned":
false,
            "AssignedPrincipalCount": 1,
            "Id": 4,
            "Name": "Infrastructure Administrators",
            "Description": "This role can view syste
m status instrumentation and system information",
            "CreatedTimestampUtc": "2019-11-07T13:1
4:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.493Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 6,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.
31Z",
        "Role": {
            "AssignedManagementGroupCount": 1,
            "HasAllDevicesManagementGroupAssigned":
true,
            "AssignedPrincipalCount": 1,
            "Id": 6,
            "Name": "Global Questioners",
            "Description": "This role can ask questi
ons and view responses for all instruction sets",
            "CreatedTimestampUtc": "2019-11-07T13:1
4:54.957Z",
            "ModifiedTimestampUtc": "2019-11-07T13:1
4:56.12Z",
            "SystemRole": true
```

Direct Consumer API call	C# code using Consumer SDK library

```
        },
        "Principal": null
    }
]
```

Now let's replace the Roles for that Principal with 2 other Roles: Custom Properties Administrators and Consumer Administrators, which have Ids 9 and 10 respectively.

Direct Consumer API call	C# code using Consumer SDK library
Making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source `[9,10]` will yield following response: **Return payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Replacing all Roles for a Principal with given Roles** Expand source `roles = connector.PrincipalRoles.ReplaceRolesForPrincipal(3, new List<int> { 9, 10 });` "roles" object will contain the same data you can see in the JSON response on the left.

```
[
    {
        "PrincipalId": 3,
        "RoleId": 9,
        "CreatedTimestampUtc": "2020-01-23T0
8:24:25.36Z",
        "Role": {
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAssi
gned": false,
            "AssignedPrincipalCount": 2,
            "Id": 9,
            "Name": "Custom Properties Admini
strators",
            "Description": "This role can ad
d, edit or delete custom properties",
            "CreatedTimestampUtc": "2019-11-0
7T13:14:55.133Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:55.133Z",
            "SystemRole": true
        },
```

Direct Consumer API call	C# code using Consumer SDK library


```
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T0
8:24:25.36Z",
        "Role": {
            "AssignedManagementGroupCount":
0,
            "HasAllDevicesManagementGroupAssi
gned": false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators
",
            "Description": "This role can man
age Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-0
7T13:14:55.523Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```


Lastly, I'll remove the assiciation to Custom Properties Administrators Role from the Principal

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 Expand source `[9]` will yield following response: **Response payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Deleting multiple Roles from a Principal** Expand source `connector.PrincipalRoles.DeleteRolesFromPrincipal(3, new List<int> { 9 });` The method will return a boolean value.

Direct Consumer API call	C# code using Consumer SDK library

```
[
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T08:24:
25.36Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned
": false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators",
            "Description": "This role can manage
Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-07T1
3:14:55.523Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```

Looking at the issue from the perspective of a Role, you can assign or unassign Principal(s) to a Role, or replace any Principals that Role has with a given set of Principals.

In the examples below we'll work with the Custom Role we've created above. It has the Id of 31 and first we'll assign 3 Principals to it:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source `[3, 5, 6]` will yield following response: **Response payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Adding multiple Principals to a Role** Expand source `principals = connector.PrincipalRoles.AddPrincipalsToRole(31, new List<int> { 3, 5, 6 });`

Direct Consumer API call	C# code using Consumer SDK library

<pre>[
    {
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-1202660629-789
336058-1343024091-23842",
            "PrincipalName": "SomeDomain\\Jane.Doe
",
            "Email": "Jane.Doe@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Jane Doe",
            "IsGroup": false
        }
    },
    {
        "PrincipalId": 5,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 1,
            "ExternalId": "S-1-5-21-1202660629-789
336158-1349024092-24850",
            "PrincipalName": "SomeDomain\\Alastor.
Rushal",
            "Email": "Alastor.Rushal@SomeDomain.co
m",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Alastor Rushal",
            "IsGroup": false
        }</pre>
	"principals" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
    },
    {
        "PrincipalId": 6,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:1
1.63Z",
        "Role": null,
        "Principal": {
            "Id": 2,
            "ExternalId": "S-1-5-21-1202660629-789
306118-1349024091-24337",
            "PrincipalName": "SomeDomain\\Marius.G
age",
            "Email": "Marius.Gage@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:
14:56.687Z",
            "ModifiedTimestampUtc": "2019-11-07T1
3:14:56.687Z",
            "SystemPrincipal": false,
            "DisplayName": "Marius Gage",
            "IsGroup": false
        }
    }
]
```

Now we will replace the existing 3 Principals with 2, one of which, "SomeDomain\\Jane.Doe" was already assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31
 	
Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source
 	
`[3, 4]`
 	
will yield following response:
 	
Return payload Expand source
 	
`[`
`  {`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Replacing all Principals for a Role with given Principals Expand source
	
	`principals = connector.PrincipalRoles.ReplacePrincipalsForRole(31, new List<int> { 3, 4 });`
	
	"principals" object will contain the same data you can see in the JSON response on the left.

```
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23
T10:00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-120266
0629-789336058-1343024091-23842",
            "PrincipalName": "SomeDomain\\
Jane.Doe",
            "Email": "Jane.Doe@SomeDomain.
com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-1
1-07T13:14:52.777Z",
            "ModifiedTimestampUtc": "2019-
11-07T13:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Jane Doe",
            "IsGroup": false
        }
    },
    {
        "PrincipalId": 4,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23
T10:00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-327632
6578-728399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\
John.Doe",
            "Email": "John.Doe@SomeDomain
",
            "Enabled": true,
            "CreatedTimestampUtc": "2016-1
1-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-
11-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "John Doe",
            "IsGroup": false
        }
```

Direct Consumer API call	C# code using Consumer SDK library

```
    }
]
```

And finally we'll remove all "SomeDomain\\John.Doe" from the Role, leaving just "SomeDomain\\Jane.Doe" assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 **Payload sent to** https://my.tachyon.server/Consumer/PrincipalRoles/Role/31 Expand source `[4]` will yield following response: **Return payload** Expand source	Use PrincipalRoles object inside the Tachyon connector instance. **Removing multiple Principals from a Role** Expand source `connector.PrincipalRoles.DeletePrincipalsFromRole(31, new List<int> { 4 });` The method will return a boolean value.

```
[
    {
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": "Jane.Doe@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Jane Doe",
            "IsGroup": false
        }
    }
]
```

As we can see, the Principal and Role assignment, unassignment and replacement endpoints mirror each other covering the approach from either Principal or Role point of view.

Lastly, you can create or remove a link between a single Principal and a single Role.

In this example we'll create a link between Customer Role, which has the id of 31 and "SomeDomain\\John.Doe" Principal, which has the Id of 4.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles
 	
Request payload sent to https://my.tachyon.server/Consumer/PrincipalRoles Expand source
 	
`{ "PrincipalId": 4, "RoleId": 31 }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "PrincipalId": 4, "RoleId": 31, "CreatedTimestampUtc": "2020-01-23T10:41:24.503Z", "Role": { "AssignedManagementGroupCount": 0, "HasAllDevicesManagementGroupAssigned": false, "AssignedPrincipalCount": 4, "Id": 31, "Name": "Custom role", "Description": "this is a description", "CreatedTimestampUtc": "2020-01-21T13:36:55.897Z", "ModifiedTimestampUtc": "2020-01-21T13:36:55.897Z", "SystemRole": false }, "Principal": { "Id": 4, "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1009", "PrincipalName": "SomeDomain\\John.Doe", "Email": "John.Doe@SomeDomain", "Enabled": true, "CreatedTimestampUtc": "2016-11-30T15:10:10.73`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Assigning Principal to a Role Expand source
	
	`var payload = new PrincipalRole { PrincipalId = 4, RoleId = 31 }; principal = connector.PrincipalRoles.Add(payload);`
	
	"principal" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
Z",
        "ModifiedTimestampUtc": "2016-11-30T15:10:10.7
3Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
}
```

and now we'll remove the link we've just created:

Direct Consumer API call	C# code using Consumer SDK library
Making a DELETE request to https://my.tachyon.server/Consumer/Role/31/Principal/4 will not yield any response payload so you should check the HTTP return code to see if the operation was successful.	Use PrincipalRoles object inside the Tachyon connector instance. **Unassigning Principal from a Role** Expand source `connector.PrincipalRoles.Delete(31, 4);` The method will return a boolean value.

Adding and removing Permissions to a Role The main method of assigning Permissions is to send two collections, one with Permissions that need to be changed or updated and another one with permissions that should be removed.

Crafting the correct payload should be done carefully as omitting an entry can result in deletion of an existing Permission.

PermissionsToSaveOrUpdate collection should have permissions you either want to add or modify. You only need to provide Ids, names are ignored. It is important to understand when the system considers given operation an 'updated' and when a 'creation' as it will behave slightly differently.

Because a Permission is between a Securable Type and a Role, with possible addition of Securable Id, those three properties define it. Operations are considered a property of a Permission and they are always considered together - i.e. a single Permission can Allow or Deny a set of Operations, but it cannot mix and match.

When you request a Permission to be added or updated, Consumer API will look at the SecurableId, SecurableTypeId and RoleId properties provided and check for any permissions already existing for this combination of values. If none are found, a new permission is added with the operations specified in the payload. If an entry is found, then Operations are examined next. If the existing Permission has Operations that aren't present in the Permission sent to the API, they are removed. Any operations present in the Permission sent to the API that do not exist in the Permission that is already in the system are added.

As you might have noticed, PermissionsToSaveOrUpdate can be used to remove Permissions by omitting Operations you want to remove. Furthermore, if you provide a Permission without any Operations, the entire Permission is removed.

Lets look at an example that should help us better understand how this functionality works.

Let us assume that we have a Permission on Instruction Set "My set" assigned to Role "Custom Role" with just one Operation "View". "My Set" has the Id of 4, "Custom Role" has the Id of 31, "Instruction Set" Securable Type has the Id of 1 and "Viewer" has the Id of 1. This means that the existing Permission will look as follows:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        }
        ]
}
```

Now let us assume that a request comes into the Consumer API that looks as follows:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 1
                        },
                                {
                                "OperationId": 3
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

This request matches a Permission we already have, because it has the same SecurableId, SecurableTypeId and RoleId, but the Operations are slightly different - Id of 3 has been added, which represents 'Questioner' Operation. Following the rules outlined above, the API will match this to the existing Permission and see that Operation with the Id of 1 already exists, so no action

is needed but the Operation with the Id of 3 doesn't exist, so it needs to be added. Afterwards we will end up with a permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        },
                {
                "OperationId": 3
        }
        ]
}
```

If now we were to make another request with with following payload:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 2
                        },
                                {
                                "OperationId": 4
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

The Consumer API will again recognise that a Permission already exists that matches the one in the payload and will examine the Operations. Because neither operation 1 nor 3, which do exist in the Permission already in the system are present in the payload, they will be removed. Operations 2 and 4 (Actioner and Approve respectively) do not exist in the Permission so they will be added and changed made will result in a Permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 2
        },
                {
                "OperationId": 4
        }
        ]
}
```

As a last example let's look at what will happen when we send this payload to the API:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": []
        }
    ],
    "PermissionsToDelete": []
}
```

Again, the system will recognise this as an existing Permission and will remove Operations 2 and 4 because they do not exist in the request payload. But since there are no Operations to add, the entire Permission will be removed.

Lastly, let's look at how an example API call would look like:

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Permissions with following payload:	Use Permissions object inside the Tachyon connector instance.
Payload sent to https://my.tachyon.server/Consumer/Permissions Expand source	**Creating Permissions** Expand source

`{`
`  "PermissionsToSaveOrUpdate": [`
`  {`
	`var payload = new Permi`
	`ssionsSaveContainer`
	`{`

Direct Consumer API call

```
            "Allowed": true,
            "SecurableTypeId": 5,
            "SecurableId": null,
            "RoleId": "31",
            "Operations": [
                {
                    "OperationId": 12
                },
                {
                    "OperationId": 13
                }
            ]
        }
    ],
    "PermissionsToDelete": []
}
```

will yield following response:

Response payload Expand source

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 5,
        "SecurableTypeName": "CustomProperty",
        "RoleId": 31,
        "RoleName": "Custom Role",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 191,
                "OperationId": 12,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
14:47:13.023Z"
            },
            {
                "PermissionId": 192,
                "OperationId": 13,
                "OperationName": "Write",
                "CreatedTimestampUtc": "2020-01-23T1
4:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-23T
```

C# code using Consumer SDK library

```
    PermissionsToSaveOr
Update = new List<Aggre
gatedPermission>
    {
        new AggregatedP
ermission
        {
            RoleId = 3
1,
            SecurableTy
peId = 5,
            Operations
= new List<PermissionOp
eration>
            {
                new Per
missionOperation
                {
                    Ope
rationId = 12
                },
                new Per
missionOperation
                {
                    Ope
rationId = 13
                }
            }
        }
    }
};
permissions = connecto
r.Permissions.AddOrUpda
te(payload);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
14:47:13.023Z"
            }
        ]
    }
]
```

2.5.9.9.3 Configuring Management Groups

Management Groups can only be assigned to a Role which has Permission to at least one Instruction Set. This prerequisite stems from internal implementation and it has to be taken into account when designing Permissions in your system.

The example below assumes that Role "Custom Role" with the Id of 31 has at least one Instruction Set Permission assigned to it.

You can assign or unassign either a single Management Group or a collection of Management Groups. To assign a single Management Group to a Role you'd issue a POST request to https://my.tachyon.server/Consumer/Roles/[role Id]/ManagementGroups /[management group id], so for instance to assign Management Group wit the Id of 7 to role with the Id of 31 you'd call https://my.tachyon.server/Consumer/Roles/31/ManagementGroups /7.

Making a DELETE request instead of POST will results in the Management Group being unassigned from the Role.

These requests do not require any payload because all information is already present in the URI. Consumer SDK equivalents are Roles.AddPermissionForManagementGroup(...) and Roles.RemovePermissionForManagementGroup(...) respectively, which simply take the Ids of the Role and the Management Group.

To assign or unassign Management Groups in bulk you have to use POST and DELETE respectively tohttps://my.tachyon.server/Consumer/Roles/ManagementGroups, and below is an example of the payload that should be sent with the request.

Direct Consumer API call	C# code using Consumer SDK library

To assign multiple Management Groups to a Role in a single call, make a POST request to https://my.tachyon.server/Consumer/Roles/ManagementGroupswith following payload:
 	
Payload sent to https://my.tachyon.server/Consumer/Roles/ManagementGroups Expand source
 	
`{`
`  "RoleId": 31,`
	Use Roles object inside the Tachyon connector instance.
	
	Assigning Management Groups to a Role Expand source
	
	`var payload = new Tachyon.SDK.Consumer.Models.Send.ManagementGroupsForRole`
	`{`
	`  RoleId = 31,`
	`  ManagementGroupIds`

Direct Consumer API call	C# code using Consumer SDK library

<pre> "ManagementGroupIds": [4,5]
}</pre>
which will yield following response:
 	
Return payload Expand source
 	
<pre>{
    "Role": {
        "Id": 31,
        "Name": "Custom role",
        "Description": "this is a description",
        "CreatedTimestampUtc": "2020-01-21T13:36:
55.897Z",
        "ModifiedTimestampUtc": "2020-01-21T13:3
6:55.897Z",
        "SystemRole": false
    },
    "ManagementGroups": [
        {
            "Id": 4,
            "Name": "Management group number 3 fr
om repository Default Inventory",
            "Description": "This is the descripti
on for Management group number 3 from repository
Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,
            "UsableId": "3",
            "HashOfMembers": "24D13054496E24C9577
4CD8E983B09C2487D60819C627750B1C16FFDEDB23DF9",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.667Z",
            "ModifiedTimestampUtc": "2020-01-08T1
6:31:02.807Z"
        },
        {
            "Id": 5,
            "Name": "Management group number 4 fr
om repository Default Inventory",
            "Description": "This is the descripti
on for Management group number 4 from repository
Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,</pre>
	<pre>= new List<int>{ 4, 5 }
	};
	role = connector.Roles.AddP
	ermissionForManagementGroup
	s(payload);</pre>
	"role" object will contain the same data you can see in the JSON response on the left.

Direct Consumer API call	C# code using Consumer SDK library

```
            "UsableId": "4",
            "HashOfMembers": "7784ED3BDA9B79F8635
E728E1E0F529200D44F3F7AABAE1A26B2ABA6DB957EDB",
            "CreatedTimestampUtc": "2019-12-02T1
2:01:21.67Z",
            "ModifiedTimestampUtc": "2020-01-08T1
6:31:02.893Z"
        }
    ]
}
```

Unassigning Management Groups from a role is a mirror operation using the DELETE verb on the same web address:

Direct Consumer API call	C# code using Consumer SDK library
To unassign multiple Management Groups from a Role in a single call, make a DELETE request to https://my.tachyon.server/Consumer/Roles/ManagementGroups with following payload: **Payload sent to** https://my.tachyon.server/Consumer/Roles/ManagementGroups Expand source `{ "RoleId": 31, "ManagementGroupIds": [4] }` This call will not return any data.	Use Roles object inside the Tachyon connector instance. **Unassigning Management Groups from a Role** Expand source `var payload = new Tachyon.SDK.Consumer.Models.Send.ManagementGroupsForRole { RoleId = 31, ManagementGroupIds = new List<int> { 4 } }; connector.Roles.RemovePermissionForManagementGroups(payload);`

2.5.9.9.4 Configuring Securable Types and Applicable Operations

We have already seen how to retrieve Securable Types and now we'll look at how we can create and modify them.

All you need to create a Securable Type is a name, which must be unique.

In order to delete a Securable Type, you have to first delete all Permissions that use that Securable Types and all Applicable Operations linked to the Securable Type.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/SecurableTypes
 	
Paylaod sent to https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Name": "Custom securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Custom securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "ModifiedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "Operations": null }`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Creating a new Securable Type Expand source
	
	`var payload = new SecurableType { Name = "Custom securable type" }; return connector.SecurableTypes.Add(payload);`
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

If you wish to update a securable type you'll have to provide its Id:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/SecurableTypes**Paylaod sent to** https://my.tachyon.server/Consumer/SecurableTypes Expand source
 	
`{ "Id": 26, "Name": "Modified securable type" }`
 	
will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 26, "Name": "Modified securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Modifying an existing Securable Type Expand source
	
	`var payload = new SecurableType { Id = 26, Name = "Modified securable type"`

Direct Consumer API call	C# code using Consumer SDK library

",
"ModifiedTimestampUtc": "2020-01-24T14:38:24.8756464Z
",
"Operations": null
}
	};
	return connector.SecurableTypes.Update(payload);
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

and to delete this Sedurable Type:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/SecurableTypes/26 This call will not yield any response.	Use SecurableTypes object inside the Tachyon connector instance. **Deleting a Securable Type** Expand source connector.SecurableTypes.Delete(26);

It is worth noting that Securable Types are created "empty" and that Applicable Operation have to be created separately.

Dealing with Operations

Applicable Operations exist only in the context of a Securable Type. While Applicable Operations must be unique within a single Securable Type, they do not have to be unique between Securable Types.

To create an Operation you have to provide either Id or Name (but not both) of Securable Type for which the Operation should be created. In the example below we'll use the Securable Type we've just created, which has the Id of 26.

Direct Consumer API call	C# code using Consumer SDK library
Make a POST request to https://my.tachyon.server/Consumer/ApplicableOperations to use Securable Type Id	Use ApplicableOperations object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library

Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{ "OperationName": "View", "SecurableTypeId": 26 }`
 	
or with this payload to use Securable Type Name:
 	
Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations Expand source
 	
`{ "OperationName": "View", "SecurableTypeName": "Modified securable type" }`
 	
Both will yield following response:
 	
Return payload Expand source
 	
`{ "Id": 63, "OperationName": "View", "SecurableTypeId": 26, "SecurableTypeName": "Modified securable type" }`
	To use Securable Type Id:
	
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation { SecurableTypeId = 26, OperationName = "View" }; operation = connector.ApplicableOperations.Add(payload);`
	
	To use Securable Type Name:
	
	Creating a new Applicable Operation Expand source
	
	`var payload = new ApplicableOperation { SecurableTypeName = "Modified securable type", OperationName = "View" }; operation = connector.ApplicableOperations.Add(payload);`
	
	"operation" object will contain the same data you can see in the JSON response on the left.

To delete an Operation you just need its Id.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/ApplicableOperations/63	Use ApplicableOperations object inside the Tachyon connector instance.

Direct Consumer API call	C# code using Consumer SDK library
	Deleting an Applicable Operation Expand source `connector.ApplicableOperations.Delete(63);`

2.6 Training Lab Guide - Using the Application Programming Interface

3 .NET Consumer SDK

.NET Consumer SDK is a self-contained C# .NET 4 assembly consisting of Consumer and Offloading SDKs.

3.1 Downloading and installing the SDK

The .NET Consumer SDK is available from the 1E Support Portal.

1. Download the version of the SDK for your version of from the .
2. Extract the files from the download zip to your choice of location.
3. Use your choice of tool or programming language, for example PowerShell or C# to create your Consumer.

For details of file versions, please refer SDK release information.

3.2 Consumer SDK library

This SDK is an object oriented abstraction of Tachyon's Consumer API. It's designed to facilitate integration of 3rd party code with Tachyon through the API.

The Consumer SDK allows virtually any managed application to access the Tachyon platform without needing the details of HTTP communication, which endpoints to use when or what input data should look like.

Tachyon consumer SDK is designed to wrap the Tachyon Consumer API, so you should be able to access every endpoint the Consumer API makes available for external callers.

You can instantiate multiple copies of the connector as the data inside it is instance based and not shared between instances.

3.2.1 SDK versioning

Each version of the SDK is designed to work with a specific version of the API. You cannot specify the API version yourself because it's hard coded into the assembly.

A mismatched SDK may result in errors or data not being sent or returned from the API.

3.2.2 TachyonConnector

Using the SDK can be as simple as instantiating a TachyonConnector object by using a constructor that requires the address of the root of the consumer API and the name of your Consumer.

This will create a connector using the DefaultTransportProxy (see below) for communicating with the Consumer API.

When you are finished with your connector, simply call Dispose on it.

3.2.2.1 TachyonConnector constructors

The connector class has 3 constructors available with the following signatures:

1. public TachyonConnector(string apiRoot, string consumerName).
2. public TachyonConnector(string apiRoot, string consumerName, ILogProxy logProxy).
3. public TachyonConnector(string apiRoot, ITransportProxy connectionProxy, ILogProxy logProxy).

Constructors 1 & 2 are almost the same, the only difference being that the 2nd one allows you to pass in a logging object.

Constructor 3 is designed to allow for full customization of the connector's behavior. It does not take Consumer name, since the transport proxy will know it already and it will provide the versioned API root address to the connector.

3.2.3 Code structure

Internally the SDK code divides into several sections:

3.2.3.1 APIs

Each class in this namespace represents a single controller from the Consumer API. Each of these classes has methods allowing you to perform every operation the controller allows.

These methods return ApiCallResponse objects.

3.2.3.2 Models

Models represent data going in and out of the Tachyon platform.

Model	Description
API	API models are high level wrappers that represent a result of an API call. The ApiCallResponse class is the object returned by every single method you can call on the API classes and contains: • Success - a boolean flag indicating call success and based on the status code returned by the HTTP call • Errors - if the call was unsuccessful you'll receive the errors in this collection • ReceivedObject - if the call was successful the object received will be in this property - note this is a generic property, its type varies depending on the called method • ResponseStatusCode - from version 3.3 inclusive, if the HTTP call has finished (for example, there has not been an exception during the call, like a timeout) this property will have the status code that was returned. **Note** Some API methods have no return value. In this instance ReceivedObject may be empty (null), have a 'string' or 'boolean' type. Please check the documentation of the method you're calling for more informtion.
Send	Contains every model representing the data you send to the API, but never receive from it. These are the classes to use when building the data to make API calls.
Receive	Contains every model representing the data you receive from the API, but not send to it. These are the classes you receive inside the ApiCallResponse.ReceivedObject.
Common	Contains models that can be either sent to or received from the API. In many cases the request and return data look exactly the same. Auxiliary classes that define expression trees, sorting and filtering expressions etc. are in here as well.

3.2.3.3 Available customization interfaces

Interface	Description
ITransportProxy	ITransportProxy defines a contract for a class to perform all of the HTTP communication with the consumer API. The Tachyon connector passes in amended (versioned) API root address to it right after it receives them and then use them for all of its HTTP calls. This interface inherits IDisposable. The TachyonConnector will call its Dispose method when it itself is being disposed of. Use this interface to fully customize HTTP communication, that is use webclient instead of httpclient. **Note** If you're using a custom implementation of this interface, take care of passing in the consumer name in the HTTP header.
IHttpClientManager	Use IHttpClientManager to customize the behavior of DefaultTransportProxy. IHttpClientManager manages the lifecycle of the HttpClient object the default proxy uses. The default client manager creates an HttpClient instance that uses default credentials. If this is not what you want, you should provide our own implementation of IHttpClientManager that will return HttpClient configured to suit your needs. **Note** The HttpClient returned by your implementation must be a fresh instance - meaning it must not have been used for making HTTP calls before. This is because the client has to be configured with the base address of the API etc. and those properties cannot be changed once even if a single HTTP call was made using the client.
ILogProxy	A rudimentary interface allowing you to receive logging from within the SDK.

3.2.3.4 Default implementations

The SDK contains default implementations for the transport proxy (class DefaultTransportProxy) and http client manager (class DefaultHttpClientManager).

DefaultTransportProxy

Default transport proxy is a class that uses a HttpClient, either created internally by DefaultHttpClientManager or created by a custom class that implements the IHttpClientManager interface and is passed in externally.

This class has two constructors:

1. public DefaultTransportProxy(ILogProxy logger, string consumerId, string apiRootAddress)
2. public DefaultTransportProxy(ILogProxy logger, string consumerId, IHttpClientManager clientManager)

The first constructor will use DefaultHttpClientManager to create a HttpClient. This means using default credentials.

The second constructor allows you to provide an object that implements IHttpclientManager, which will provide (and dispose of) the HttpClient object.

DefaultHttpClientManager

Default http client manager is a simple class that creates an HttpClient that uses default credentials and simply disposes of it when told to do so.

3.2.3.5 Tools

ErrorCreator

Error creator is a helper class containing two static methods allowing you to convert either HttpResponseMessage or an Exception to an Error object that you would receive in an ApiCallResponse object.

3.2.4 Code examples

3.2.4.1 Example 1: Access Tachyon using default credentials

Creating default connector

```
TachyonConnector connector = new TachyonConnector("http://myserver.mydomain.com
/consumer", "myConsumer", null);
```

3.2.4.2 Example 2: Access Tachyon using specific credentials

Custom Http manager class

```
class MyManager : IHttpClientManager
{
        private readonly string userName;
    private readonly string password;

        public MyManager(string userName, string password)
        {
                this.userName = userName;
        this.password = password;
        }

        public System.Net.Http.HttpClient GetClient()
```

```
    {
                HttpClientHandler httpMessageHandler = new HttpClientHandler
        {
                UseDefaultCredentials = false,
                Credentials = new NetworkCredential(this.userName, this.passwor
d)
        };
        return new HttpClient(httpMessageHandler);
        }

        public void DisposeOfClient(System.Net.Http.HttpClient client)
    {
                client.Dispose();
        }
}
```

Creating connector

```
IHttpClientManager clientManager = new MyManager("username goes here", "passwor
d goes here");
ITransportProxy proxy = new DefaultTransportProxy(null, "myConsumer", clientMan
ager);
TachyonConnector connector = new TachyonConnector("http://myserver.mydomain.com
/consumer", proxy, null);
```

3.2.4.3 Example 3: Access Tachyon using a custom transport proxy

Creating a connector with custom transport proxy

```
class myConnectionProxy : ITransportProxy
{
        [...]
}

[...]

var myConnectionProxyInstance = new myConnectionProxy([...]);
TachyonConnector connector = new TachyonConnector("http://myserver.mydomain.com
/consumer", myConnectionProxyInstance, null);
```

3.2.4.4 Example 4: A complete example

A complete SDK example Expand source

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Tachyon.SDK.Consumer;
```

```
using Tachyon.SDK.Consumer.Enums;

namespace Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            var connector = new TachyonConnector("http://localhost/consumer", "
MyCustomConsumer", null);
            var devices = connector.Devices.GetDevices();
            if (devices.Success)
            {
                foreach (var device in devices.ReceivedObject)
                {
                    Console.WriteLine("Name: {0} - Os: {1}", device.Fqdn, devic
e.OsVerTxt);
                }
            }

            var instructionDefinitions = connector.InstructionDefinitions.Get(n
ew List<InstructionType> {InstructionType.Question});
            if (instructionDefinitions.Success)
            {
                foreach (var instructionDefinition in instructionDefinitions.Re
ceivedObject)
                {
                    Console.WriteLine(instructionDefinition.Name);
                }
            }

            var definition = connector.InstructionDefinitions.GetInstructionDef
inition("1E-Explorer-TachyonCore-DisabledServices");
            if (definition.Success)
            {
                var def = definition.ReceivedObject;
                Console.WriteLine("Name: {0}", def.Name);
                Console.WriteLine("Description: {0}", def.Description);
                Console.WriteLine("Instruction TTL: {0}", def.InstructionTtlMin
utes);
                Console.WriteLine("Responses TTL: {0}", def.ResponseTtlMinutes)
;
                Console.WriteLine("Schema:");
                foreach (var schemaColumn in def.Schema)
                {
                    Console.WriteLine("   Field Name: {0}. Data Type: {1}", sch
emaColumn.Name, schemaColumn.Type);
                }
                Console.WriteLine("Aggregation:");
                foreach (var aggregationColumn in def.Aggregation.Schema)
```

```
                {
                    Console.WriteLine("   Field Name: {0}. Data Type: {1}", agg
regationColumn.Name, aggregationColumn.Type);
                }
            }

            var myDefinition = connector.InstructionDefinitions.GetInstructionD
efinition("1E-Explorer-TachyonCore-DisabledServices");
            if (myDefinition.Success)
            {
                var result = connector.Instructions.SendInstruction(myDefinitio
n.ReceivedObject.Id, null, 60, 120);
                if (result.Success)
                {
                    Console.WriteLine("successfully issued an instruction");
                    Thread.Sleep(2000);
                    var answers = connector.Responses.GetAllAggregatedResponses
(result.ReceivedObject.Id);
                    if (answers.Success)
                    {
                        foreach (var answer in answers.ReceivedObject.Responses
)
                        {
                            Console.WriteLine("Caption: {0} - Count :{1}",answe
r.Values["Caption"], answer.Values["Count"]);
                        }
                    }
                }
            }
        }
    }
}
```

3.3 Offloading SDK library

This SDK has been created to simplify creation of software that will function as an offloading client, receiving responses from Tachyon.

The Offloading SDK provides a managed DLL for use in an ASP.NET application. This helps create a Tachyon offloading target. The DLL contains a definition for the class encapsulating the responses received from Tachyon. It also includes a code for unpacking and decompressing the information Tachyon sends.

3.3.1 Creating an Offloading Target

The OffloadTarget needs to accept HTTP POSTs. If coded in .NET, it would typically be a WebApi application.

The DLL provided by the SDK is named **Tachyon.SDK.Offloading.dll**. After referencing the DLL from the WebApi project and importing the **Tachyon.SDK.Offloading** namespace, an action method that receives the posts can be as short as this:

Offloading target

```
        public HttpResponseMessage Post([CustomBody]byte[] responses)
        {
            if (responses == null)
            {
                return new HttpResponseMessage(HttpStatusCode.BadRequest);
            }

            ResponseRaw[] responseRaws = JsonResponseDeserializer.DeserializeOb
ject(responses);
            ProcessResponses(responseRaws);
            return new HttpResponseMessage(HttpStatusCode.Accepted);
        }
```

The most important line is:

ResponseRaw

```
    ResponseRaw[] responseRaws = JsonResponseDeserializer.DeserializeObject(re
sponses);
```

- **ResponseRaw** - defined in the SDK DLL encapsulating raw responses offloaded from Tachyon
- **JsonResponseDeserializer.DeserializeObject** - converts bytes received into raw responses, and handles decompression as required
- **[CustomBody]** - captures raw posted data so it can be passed into DeserializeObject for decompression

For an explanation of why **[CustomBody]** is needed and why the default WebApi binding won't work as expected, read:

Accepting Raw Request Body Content with ASP.NET Web API - https://weblog.west-wind.com/posts/2013/Dec/13/Accepting-Raw-Request-Body-Content-with-ASPNET-Web-API.

In the above example, the method **ProcessResponses** would contain the custom code that processes the data received from Tachyon.

3.4 SDK release information

3.4.1 .NET Consumer SDK 8.1 release notes – last updated 18-August-2022

These notes contain late-breaking information about needed to successfully install and use the SDK. For the most up-to-date information, please see the online readme. You should thoroughly read these release notes and the online documentation for the SDK before installing the software.

3.4.1.1 Online help

has online help that shows you how to install, configure and use the product. Click the following link to view the help:

- Integrating with - using the API and SDK.

- .NET Consumer SDK - downloading and installing.

For support queries, please:

- Use where you can search the KB or submit a support case.
- Contact support@1e.com.

3.4.2 Tachyon .NET Consumer SDK release history

Tachyon .NET Consumer SDK:

File name	Description	8.1	8.0	5.2	5.1	5.0	4.1
Newtonsoft.Json.dll	helper library for handling JSON data (https://www.newtonsoft.com/json)	13.0.1	9.0.1				
Tachyon.SDK.Consumer.dll	Consumer SDK library	8.1.2.102	8.0.4.237	5.2.5.181	5.1.0.663	5.0.0.592	4.1.0.206
Tachyon.SDK.Consumer.XML	Consumer SDK Xml documentation for Intellisense support	n/a					

Tachyon Offloading SDK:

File name	Description	8.1	8.0	5.2	5.1	5.0	4.1
Newtonsoft.Json.dll	helper library for handling JSON data (https://www.newtonsoft.com/json)	13.0.1	9.0.1				
System.Net.Http.Formatting.dll	helper library for handling HTTP formatting	5.2.30128					
System.Web.Http.dll	helper library for making HTTP web calls	5.2.30128					
Tachyon.SDK.Offloading.dll	Offloading SDK library	8.1.2.102	8.0.4.237	5.2.5.181	5.1.0.663	5.0.0.592	4.1.0.206

Versions

- Web Release-8.1 (04-May-2022) - Tachyon .NET Consumer SDK 81.2.102
- Web Release-8.0 (24-Nov-2021) - Tachyon .NET Consumer SDK 8.0.4.237
- Web Release-5.2 (21-May-2021) - Tachyon .NET Consumer SDK 5.2.5.181
- Web Release-5.1 (25-Sep-2020) - Tachyon .NET Consumer SDK 5.1.0.663
- Web Release-5.0 (04-May-2020) - Tachyon .NET Consumer SDK 5.0.0.592
- Web Release-4.1 (22-Nov-2019) - Tachyon .NET Consumer SDK 4.1.0.206