PUSH TECHNOLOGY™

Diffusion 6.7 User Guide

Contents

List of Figures **10**

List of Tables **12**

Part I: Welcome **15**

Introducing Diffusion **16**

Introducing topics and data 16

Introducing sessions 20

What's new in Diffusion 6.7? **23**

Part II: Quick Start Guide **25**

Part III: Design Guide **26**

Support **27**

System requirements for the Diffusion server 27

Platform support for the Diffusion API libraries 30

Feature support in the Diffusion API 32

License types 37

Protocol support 38

Browser support 39

Browser limitations 40

WebSocket limitations 40

Cross-origin resource sharing limitations 40

Browser connection limitations 41

Designing your data model **42**

Topic tree 42

Topic naming 43

Topic selectors 44

Topic views 50

Session trees 67

Topics 69

Properties of topics 69

JSON topics 74

Binary topics 76

String topics 78

Int64 topics ... 80
Double topics ... 82
Time series topics ... 83
Routing topics ... 86
Slave topics (DEPRECATED) ... 87
RecordV2 topics ... 89
Pub-sub ... 93
Publishing data ... 94
Subscribing to topics ... 96
Topic notifications ... 98
Request-response messaging ... 100
Conflation ... 102
Using conflation ... 102
Designing your solution ... 103
Servers ... 104
Server clusters for high availability ... 105
Session replication ... 107
Topic replication ... 111
Failover of active update sources ... 112
Sharing data with remote servers ... 113
Remote topic views ... 115
Fan-out ... 116
Using missing topic notifications with fan-out ... 118
Topic persistence ... 120
Topic delivery priority ... 121
Automatic topic removal ... 122
Clients ... 123
Client types ... 123
Using clients ... 124
Using clients for control ... 124
Client coordination ... 127
User-written components ... 127
Third party components ... 129
Load balancers ... 129
Web servers ... 130
JMS ... 133
Example solutions ... 134
Example: Simple solution ... 135
Example: A solution using clients ... 136
Example: Scalable and resilient solution ... 137
Security ... 137
Role-based authorization ... 138
Permissions ... 142
Pre-defined roles ... 147
Authentication ... 149
User-written authentication handlers ... 153
System authentication handler ... 154
Pre-defined users ... 155
Topic ownership ... 156
DEPRECATED: Authorization handlers ... 157

Part IV: Developer Guide ... 160

Best practice for developing clients ... 162
Feature support in the Diffusion API ... 163

Client basics **168**
JavaScript 168
Apple 172
Android 174
Java 176
.NET 179
C 180
Python 185
Connecting to the Diffusion server **186**
Connecting basics 188
Connecting securely 194
Connecting to the Diffusion server with a security principal and credentials 196
Connecting through an HTTP proxy 199
Connecting through a load balancer 202
Reconnect to the Diffusion server 203
Detecting connection problems 206
Specifying a reconnection strategy 206
Session failover 216
Pinging the Diffusion server 218
Change the security principal and credentials associated with your client session 219
Session properties 220
Session filtering 222
Receiving data from topics **224**
Subscribing to topics 225
Using streams for subscription 227
Fetching the current value of a topic 233
Receiving topic notifications 240
Managing topics **244**
Adding topics with topic specifications 245
Example: Create a JSON topic 245
Handling subscriptions to missing topics 254
Example: Receive missing topic notifications 256
Defining a recordV2 schema 266
Update recordV2 topics 270
Subscribe to recordV2 topics 275
Removing topics 283
Removing topics automatically 284
Receiving topic notifications 287
Updating topics **291**
Session locks 293
Using time series topics **297**
Example: Publish a time series 300
Example: Subscribe to a time series 305
Managing subscriptions **309**
Example: Subscribe other clients to topics 311
Example: Receive notifications when a client subscribes to a routing topic 318
Using request-response messaging **323**
Sending request messages to a message path 325
Sending request messages to a session 331
Sending request messages to a session filter 338
Authenticating new sessions **342**
Example: Register an authentication handler 343
Developing a control authentication handler 352
Updating the system authentication store **356**
DSL syntax: system authentication store 356

Example: Update the system authentication store....360
Updating the security store....367
DSL syntax: security store....368
Example: Update the security store....370
Managing sessions....377
Working with session properties....378
Handling client queues....383
Flow control....384
Configuring conflation....385
Logging from the client....386
Logging in JavaScript....386
Logging in Apple....387
Logging in Android....388
Logging in Java....388
Logging in .NET....390
Logging in C....391
Developing other components....391
Local authentication handlers....391
Developing a local authentication handler....391
Using Maven to build Java Diffusion applications....393
Build client applications....394
Build server application code with Maven....394
Testing....395
Benchmarking suite....395
Creating a remote server definition....396

Part V: Administrator Guide....397

Installing the Diffusion server....398
System requirements for the Diffusion server....398
Using the Diffusion server installer in graphical mode....401
Using the Diffusion server installer in headless mode....403
Installing the Diffusion server using Red Hat Package Manager....404
Installing the Diffusion server using Docker....405
Next steps with Docker....406
The Diffusion license....407
License restrictions....408
Updating your license file....409
Installed files....410
Verifying the Diffusion installation....412
Configuring your Diffusion server....414
XML configuration....414
Obfuscation tool....416
Programmatic configuration....417
Using the Server API for configuration....417
Configuring the Diffusion server....418
Configuring fan-out....418
Configuring authentication handlers....420
Configuring performance....422
Configuring topic persistence....422
Server.xml....423
Configuring connectors....436
Connectors.xml....438
Configuring security....446
Security.store....446

SystemAuthentication.store ... 449
Securing the console ... 453
Configuring logging on the Diffusion server ... 454
Configuring log4j2 ... 455
Log4j2.xml ... 456
Configuring legacy logging ... 457
Logs.xml ... 458
Logging using another SLF4J implementation ... 461
Configuring JMX ... 462
Configuring the Diffusion JMX connector server ... 462
Configuring a remote JMX server connector ... 463
Configuring a local JMX connector server ... 464
Management.xml ... 465
Configuring replication ... 466
Configuring the Diffusion server to use replication ... 466
Configuring the Hazelcast datagrid ... 467
Replication.xml ... 469
Configuring the Diffusion web server ... 472
Configuring Diffusion web server security ... 473
WebServer.xml ... 474
Aliases.xml ... 479
ConnectionValidationPolicy.xml ... 480
Env.xml ... 481
Mime.xml ... 482
Statistics.xml ... 482
Cross domain ... 484
Starting the Diffusion server ... 484
Running from within a Java application ... 485
Network security ... 487
Going to production ... 490
Pre-production testing ... 490
Setting up your test environment ... 490
Understanding production usage conditions ... 491
Types of testing ... 494
Testing your security ... 495
Planning for production ... 496
Deploying to your production environment ... 496
Tuning ... 497
Concurrency ... 497
Buffer sizing ... 499
Message sizing ... 500
Client queues ... 501
Client multiplexers ... 501
Connectors ... 502
Thread pools ... 502
Session reconnection ... 505
Client failover ... 507
Java memory usage ... 508
Platform-specific issues ... 508
Socket issues ... 508
Managing and monitoring your running Diffusion server ... 510
JMX ... 511
Using Java VisualVM ... 512
Using JConsole ... 514
MBeans ... 517

The JMX adapter....528
Metrics....531
Configuring metrics....535
Diffusion management console....538
Configuring the Diffusion management console....540
Logging....542
Logging back-end....543
Logging reference....544
Log messages....547
Connection counts....614
Integration with Splunk....615
Web servers....618
Diffusion web server....618
Server-side processing....619
Hosting a status page on the Diffusion web server....620
Hosting Diffusion web clients in a third-party web server....620
Running the Diffusion server inside of a third-party web application server....621
Example: Deploying the Diffusion server within Tomcat....622
Other considerations when running the Diffusion server inside of a third-party web application server....624
Cross domain policies....625
Load balancers....625
Routing strategies at your load balancer....626
Monitoring available Diffusion servers from your load balancer....628
Compositing URL spaces using your load balancer....628
Secure Sockets Layer (SSL) offloading at your load balancer....629
Using load balancers for resilience....629
Common issues when using a load balancer....630
MQTT support....631
Using MQTT....632
Configuring MQTT....634
Kafka adapter....635
Configuring the Kafka adapter....635
Running the Kafka adapter....638
JMS adapter....639
Configuring the JMS adapter....640
Running the JMS adapter....640
CDC database adapter....640
Configuring the CDC adapter....642
Running the CDC adapter....643
Demos....644
Demos....644

Part VI: Upgrading Guide....646

Interoperability....647
Upgrading from version 5.x to version 6.0....648
Upgrading from version 6.0 to version 6.1....654
Upgrading from version 6.1 to version 6.2....658
Upgrading from version 6.2 to version 6.3....661
Upgrading from version 6.3 to version 6.4....664
Upgrading from version 6.4 to version 6.5....666
Upgrading from version 6.5 to version 6.6....670
Upgrading from version 6.6 to version 6.7....673
Upgrading to a new patch release....676

Chapter : Appendices.....677

Appendix A: Document conventions..... 678

Appendix B: Glossary.....679

A..... 680
C..... 680
D..... 682
E..... 683
F.....684
G..... 684
H.....685
I..... 685
J.....686
L.....687
M.....688
N.....688
P..... 689
Q.....691
R..... 691
S..... 692
T..... 694
U.....695
V..... 696
W..... 696
X..... 696

Appendix C: Trademarks.....698

Appendix D: Copyright Notices.....700

ANTLR..... 702
Bouncy Castle..... 702
Apache Commons Codec..... 702
Apache Portable Runtime..... 702
Bootstrap.....703
CQEngine..... 703
cron4j..... 703
d3..... 703
disruptor..... 703
Fluidbox..... 704
GeoIP2 API..... 704
GeoLite2 City Database..... 704
GeoIP2 API..... 704
geronimo-jms_1.1_spec..... 704
Google code prettify.....705
hashmap.....705
Hazelcast..... 705
HPPC..... 706
htmlcompressor.....706

inherits....706
iStack Common Runtime....707
jackson-annotations....707
jackson-core....707
jackson-dataformat-cbor....707
jackson-databind....707
JAXB....707
JCIP Annotations....708
JCTools....708
jQuery....708
jquery.floatThead....708
json-simple....708
Knockout....709
libwebsockets....709
License3j....709
log4j2....709
loglevel....710
long....710
Metrics....710
Minimal JSON....711
Modernizr....711
NLog....711
opencsv....711
OpenSSL....711
PCRE....711
Picocontainer....712
Prometheus Java Simpleclient....712
Rickshaw....713
Servlet API....713
SLF4J....713
slf4j-android-logger....713
SocketRocket....713
streamsupport....714
Tabber....714
Tapestry (Plastic)....714
when....715
ws....715
Licenses....715
Apache License 2.0....715
BSD 3-clause License....718
Common Development and Distribution License....719
Eclipse Public License – v 1.0....723
GNU General Public License, version 2, with the Classpath Exception....726
ISC License –....730
The GNU Lesser General Public License, version 2.1 (LGPL-2.1)....731
The MIT License (MIT)....736
OpenSSL and SSLeay Licenses....737

List of Figures

Figure 1: Example topic tree..... 42

Figure 2: Pub-sub model..... 94

Figure 3: A client session registers a handler on part of the topic tree..... 101

Figure 4: A client session can send requests through a message path to a known client session..... 101

Figure 5: A client can send requests through a message path to a set of client sessions..... 101

Figure 6: Information sharing using a datagrid..... 105

Figure 7: Session replication..... 108

Figure 8: Topic replication..... 111

Figure 9: Fan-out..... 116

Figure 10: Missing topic notification propagation..... 119

Figure 11: Using a web server with Diffusion..... 130

Figure 12: Deploying Diffusion inside a web application server..... 131

Figure 13: A simple solution..... 135

Figure 14: Clients for different purposes..... 136

Figure 15: Architecture using replication and fan-out..... 137

Figure 16: Authentication process for clients..... 151

Figure 17: Session state model..... 186

Figure 18: Flow of requests and responses when connecting to Diffusion through a proxy..... 199

Figure 19: A stream....225
Figure 20: Automatic topic removal, topic views and fan-out....287
Figure 21: Data type hierarchy....324
Figure 22: Connecting to Diffusion JMX....511
Figure 23: Java VisualVM: Overview tab....513
Figure 24: JConsole New Connection dialog: Remote Process....514
Figure 25: JConsole New Connection dialog: Remote Process....515
Figure 26: JConsole New Connection dialog: Local Process....516
Figure 27: The server MBean stopController operation showing in JConsole....518
Figure 28: Reflecting MBeans as topics....529
Figure 29: Showing a composite attribute as a topic nest....530
Figure 30: Topics reflecting an ArrayType MXBean attributes....531
Figure 31: The new console Overview tab....539
Figure 32: Welcome tab of the Splunk web UI....616
Figure 33: The Splunk Set source type dialog....616
Figure 34: The Data Preview panel....617
Figure 35: The Splunk search summary panel....617
Figure 36: Sticky-IP in F5 BIG-IP....627

List of Tables

Table 1: Supported platforms and transport protocols for the client libraries..... 30
Table 2: Capabilities provided by the Diffusion client libraries..... 32
Table 3: License capabilities..... 37
Table 4: Supported protocols by client..... 38
Table 5: Supported browsers..... 39
Table 6: Support for WebSocket..... 40
Table 7: Support for CORS..... 40
Table 8: Maximum supported connections..... 41
Table 9: Types of topic selector..... 44
Table 10: Descendant pattern qualifiers..... 45
Table 11: Reference topic property mapping..... 60
Table 12: Reference topic type mapping..... 64
Table 13: Branch mapping example..... 68
Table 14: Properties available for topics of each type..... 69
Table 15: Data types for schema fields..... 92
Table 16: Notification types..... 99
Table 17: Supported protocols by client..... 124
Table 18: List of path-scoped permissions..... 142
Table 19: List of global permissions..... 146

Table 20: Client operations that require authentication.....153
Table 21: Types of authentication handler.....153
Table 22: Authorization handler methods.....157
Table 23: Capabilities provided by the Diffusion client libraries.....163
Table 24: Supported platforms and transport protocols for the client libraries.....170
Table 25: Supported platforms and transport protocols for the client libraries.....173
Table 26: Supported platforms and transport protocols for the client libraries.....175
Table 27: Supported platforms and transport protocols for the client libraries.....177
Table 28: Supported platforms and transport protocols for the client libraries.....179
Table 29: Supported platforms and transport protocols for the client libraries.....181
Table 30: Session filter search clause operators.....222
Table 31: Session filter boolean operators.....223
Table 32: Removal condition types.....284
Table 33: Time series event metadata.....297
Table 34: Conflation topic properties.....386
Table 35: Log levels.....386
Table 36: Log levels.....387
Table 37: Log levels.....388
Table 38: Log levels.....388
Table 39: Artifacts.....393
Table 40: Installed files.....410
Table 41: Tools and utilities.....411
Table 42: XML Value types.....414
Table 43: Connectors properties.....437
Table 44: Values that can be configured for a thread pool.....503
Table 45: Events that a thread pool notification handler can act on.....504
Table 46: Notifications as topics.....529

Table 47: Metrics provided by Diffusion....532
Table 48: Log levels....544
Table 49: Fields included in the logs....545
Table 50: Examples of routing strategies....626
Table 51: Subscribing (updates from Diffusion to MQTT client)....633
Table 52: Publishing (updates from MQTT client to Diffusion)....634
Table 53: Demos provided with the Diffusion server....644
Table 54: API interoperation....647
Table 55: API features removed in version 6.0....649
Table 56: API features deprecated in version 6.0....651
Table 57: API features removed in version 6.1....655
Table 58: API features deprecated in version 6.1....656
Table 59: API features removed in version 6.2....659
Table 60: API features deprecated in version 6.2....659
Table 61: API features deprecated in version 6.3....662
Table 62: API features deprecated in version 6.4....664
Table 63: API features removed in version 6.4....665
Table 64: API features deprecated in version 6.5....667
Table 65: API features removed in version 6.5....667
Table 66: API features deprecated in version 6.6....671
Table 67: API features removed in version 6.6....671
Table 68: API features deprecated in version 6.7....674
Table 69: API features removed in version 6.7....675
Table 70: Typographic conventions used in this manual....678

Part I

Welcome

Welcome to the Push Technology User Manual for Diffusion™

The manual is regularly updated, but if you require further help, see the articles and forums in our Support Center: https://www.pushtechnology.com/support/.

New to Diffusion?

- Learn what Diffusion is and what it can do for your organization: Introduction
- Get started with Diffusion: Quick Start Guide

Ready to start building your Diffusion solution?

- Decide what your Diffusion solution will look like: Design Guide on page 26
- Develop your Diffusion clients: Developer Guide on page 160
- Set up and manage your Diffusion server and solution: Administrator Guide on page 397

About to upgrade from an earlier version of Diffusion?

- See what's new in the latest version of Diffusion: What's new in Diffusion 6.7? on page 23
- Check how changes might affect your existing Diffusion solution: Upgrading

In this section:

- Introducing Diffusion
- What's new in Diffusion 6.7?

Introducing Diffusion

The Diffusion Intelligent Data Platform from Push Technology is designed to manage, distribute and synchronize data.

Diffusion provides powerful data distribution capabilities, allowing applications to publish real-time data to large numbers of web, mobile, and IoT client devices.

Diffusion helps you deal with the challenges of limited bandwidth, unreliable network connectivity and high update volume, with features like delta updating and data conflation.

This introduction explains the high-level concepts behind Diffusion.

Introducing topics and data

Diffusion stores and distributes data through topics.

Topics

At the heart of the Diffusion model lies the concept of a topic. This page covers the various aspects of topic management that make Diffusion unique, including persistent subscriptions using topic selectors; topic query capabilities; automatic topic removal; how Diffusion achieves high network efficiency using delta streaming, conflation, and compression; and how to secure topic data.

In Diffusion, data is stored and distributed through topics. Each topic has a topic type and a current data value which is maintained in memory on the server. A topic's type determines the data values that can be stored and published through the topic.

Granted sufficient security privileges, a client session can subscribe to a topic to receive notifications when the topic value changes and can also update the value. When a topic is updated, all its subscribers are notified of the new value. Diffusion takes care of efficiently broadcasting value changes, even if there are many thousands of subscribers.

Topics are identified by topic paths. A topic path is a string of parts separated by the / character, for example, weather/capitals/athens. Together, the set of topic paths forms the topic tree.

The topic tree allows topics to be addressed in groups using special expressions called topic selectors. For example, the topic selector ?weather/capitals/ can be used to subscribe to all topics below the topic path weather/capitals. See the full syntax of topic selector expressions.

Topics are lightweight and cheap to create and destroy. There are commercial Diffusion applications that use millions of topics hosted in a single server and create tens of thousands of topics when a new tranche of data items becomes available. The low cost per topic allows for topic trees with a fine-grained mapping to logical data models, with each topic representing a discrete data item that can be updated independently.

Topic types, values, and updates

There are nine topic types that can be grouped into four categories: primitive; composite; multi-valued; and reference.

The four primitive topic types — string, int64, double, and binary — are used for topics with simple, atomic values. String topics store text, int64 and double topics store numbers, and binary topics can store arbitrary data such as a PNG image.

There are two composite topic types: JSON and recordV2. A JSON topic has a JSON value, a format that is familiar to developers and easy for JavaScript clients to process. RecordV2 topics store an array of fields, constrained by an optional schema. RecordV2 topics exist as an upgrade path for applications that were previously using the removed record topic type – new applications should use JSON topics.

Many applications can get by using only the primitive and JSON topic types. Multi-valued and reference topic types are more specialized and less commonly used.

There is a single multi-valued topic type. The time series topic stores a history of events. Events are created by a special type of update. Each event has a timestamp, records the security principal that created it, and has a value. The values for a time series topic are all of the same type, which can be string, int64, double, binary, JSON, or recordV2 – that is, the same data types used for primitive and composite topics. Ranges of events can be queried by data range or event offset.

There are two reference topic types: slave and routing. These are quite different from other topic types. Rather than storing a data value, they re-present the values of other source topics at their topic path. A source topic can be any primitive, composite, or multi-valued topic. A slave topic has a fixed source topic. A routing topic calls out to an application-provided routing handler to determine the source topic for each subscribing session.

When a topic is created, it has no value. A client session can update the topic by providing a value. Primitive and composite topics store the latest received value. Time series topics store a configurable history of values. Each reference topic re-presents the value or values of its source topic.

Sometimes there is no need to store the current value of a topic. Perhaps the value has a limited lifetime and is only of transient worth. A topic can be configured not to retain its last value to reduce the server memory footprint. However, this disables the delta streaming optimization (see below), so is not often done.

For a given topic, the order of value updates is preserved from the source session to the subscriber sessions. If a session updates a string topic with the value *A1* followed by *A2*, the server will notify subscribing sessions of the updates in that order. No guarantees are made about the order of updates across topics. For example, if a session updates topic A with the values *A1* and *A2*, and topic B with the values *B1* and *B2* , one subscriber might receive *A1*, *B1* , *B2* , *A2*, and another might receive *B1* , *B2* , *A1*, *A2*.

Adding and removing topics

Sessions with appropriate security privileges can add and remove topics. The topic path and topic specification are supplied when adding a topic. The topic specification consists of the topic type and a set of topic properties that allow the behavior of the topic to be adapted to application needs. Some topic properties are specific to the topic type. For example, the *TIME_SERIES_EVENT_VALUE_TYPE* property configures the data type of the values for a time series topic, and the *SCHEMA* property configures an optional schema for recordV2 topics.

All topic types support the optional *REMOVAL* property, which configures an automatic removal policy. Each policy provides a set of conditions under which the server will remove a topic. You can configure a topic to be removed at a future time, if it has stopped receiving updates, if it has no subscribers, or when the server has no client sessions matching specific criteria. The criteria are expressed in terms of session property values.

Subscribing to topics

The server maintains a real-time data model, presented through topics. Each client selectively subscribes to a subset of the data model, according to the needs of the application and data security restrictions applied at the server. Topics provide a fine-grained mapping of the logical data model, so in a typical application each client has a unique partial view of the data model. The client library retains the values of each subscribed topic. The server sends updates to keep the client's view synchronized.

Client sessions subscribe to topic paths using topic selectors. The server persists the set of topic selectors for each session, and dynamically joins selectors against its topics to resolve subscriptions. The dynamic join between topic selectors and topics is unique to Diffusion and is a powerful way to link client applications with a changing data model. The set of topic selectors defines the view of the data model the client is interested in. The server keeps each session up-to-date with the available data that matches the provided topic selectors. Let's look at how this works.

When a session subscribes with a topic selector, the server will resolve subscriptions to all topics with paths matching the topic selector. The session will be notified of the resolved subscriptions and the current value of each topic that has one. The subscription notification includes the topic specification. The server will further notify the session of topic value changes as they occur. A session can subscribe to a topic path for which there is no topic. If a topic is created for the path at a later time, the server will resolve the subscription and notify the session. For example, if a topic weather/capitals/paris is added, subscriptions will be resolved for all sessions that have previously subscribed using the topic selector ? weather/capitals/ . The server will notify the subscribing sessions of the new subscriptions.

If the topic is removed, any resolved subscriptions will be removed, and the previously subscribed sessions will be notified of the unsubscription.

A session can unsubscribe from paths using a topic selector. Subscriptions will be removed for any topics matching the selector to which the session was previously subscribed, and the server will notify the session of the unsubscription. Like subscribe requests, unsubscribe requests are persisted by the server. The session's selector set is the accumulation of the subscribe and unsubscribe events, in the order received. For example, if a session subscribes to ?weather/capitals/ and unsubscribes from >weather/capitals/athens, the selector set will match all topics below weather/capitals except for weather/capitals/athens. On the other hand, if a session first unsubscribes from >weather/capitals/athens and then subscribes to ?weather/capitals/, the subscription will mask the more specific, earlier unsubscription and the selector set will match all topics below weather/capitals.

The dynamic joins extend to slave and routing reference topics. Subscriptions to reference topics are only resolved if the referenced source topic also exists. Subscriptions to reference topics will be removed if either the reference topic or the source topic is removed.

Fetching topic data

A session can fetch the topic specifications and current values of a set of topics. This is a one-off operation that captures a snapshot of the data – the session is not notified of later value updates – but is useful for applications needing to present a static view of the available data.

The set of topics to fetch is specified with a topic selector and can be further constrained to allow the topic tree to be explored page-by-page.

How Diffusion makes efficient use of the network

Many aspects of Diffusion across different architectural layers combine to allow very efficient delivery of application data over the network. The performance translates directly into tangible financial savings for Diffusion users and their customers – more application data can be streamed using less network bandwidth. In addition, applications can provide richer and more data-intensive views.

Diffusion uses a proprietary binary network protocol, designed with close attention to minimizing transport framing costs. For each session, the server balances the batching of updates into network operations against their timely delivery.

The fine-grained mapping of topics to the logical data model allows an application client to select only the data items that it needs. The server maintains the topic selectors for each session, so can immediately subscribe them to new data items without additional interactions. In contrast, publish-and-subscribe messaging systems often require applications to publish the availability of a new data item on one channel, and for interested clients to respond to this event by individual subscribing, which is expensive to process and introduces unnecessary delays.

Through the subscription-based approach, each client session is synchronized with the topics it is subscribed to. Consequently, the server only needs to inform each client of a topic's path and specification when the subscription is resolved. Even better, it allows changes to a topic's value to be sent as an optimal delta stream.

A delta stream encodes a change to the value by sending only the differences between the old value and the new value. Updates to values frequently only affect part of the value. Consider a JSON value – typically the structure of the value including object keys, white space, and delimiters is unchanged between successive updates. Delta streaming is performed automatically and is transparent to the application. The server calculates the differences between the previous value and the new value and sends this to the client. The client applies the differences to its copy of the previous value to calculate the new value. Delta streams are also used when a client session uses an update stream to send a sequence of updates to a topic. Again, Diffusion automatically and transparently calculates and sends differences between successive values. The synchronized, stateful communication used by Diffusion is much more network efficient than the stateless communication used by messaging-based or polling-based systems.

Topic value updates sent from the server to sessions are compressed and decompressed by the clients. The server compresses each update once and re-uses the result for all of the subscribers. Compression is complementary to delta streaming and provides additional efficiency benefits.

Diffusion's conflation feature improves the efficiency, reliability, and timeliness of topic updates sent to slow or temporarily disconnected sessions. The server has a queue of updates for each session. Updates can back-up on a queue if the session is temporarily disconnected, there is a network bottleneck, or the client application is performing slowly. Conflation addresses the backlog by selectively removing out-of-date topic updates. This reduces server memory footprint and the amount of network data required to bring a session back up to date. A conflation policy can be tuned for each topic using the *CONFLATION* topic property.

Controlling access to topic data

Using Diffusion's role-based authorization system, individual sessions can be granted or denied the rights to add and remove a topic, to subscribe using a topic selector, to view a topic value, or to update a topic value.

Each session has a set of roles obtained through the authentication process or set by control sessions. Each role grants a session various security permissions. Access to topics is controlled via the topic permissions *MODIFY_TOPIC*, *READ_TOPIC*, *SELECT_TOPIC*, and *MODIFY_TOPIC*. Time series topics can be further controlled by the topic permissions *QUERY_OBSOLETE_TIME_SERIES_EVENTS*, *EDIT_TIME_SERIES_EVENTS*, and *EDIT_OWN_TIME_SERIES_EVENTS*, which grant sessions additional control over the history of time series events.

Topic permissions are assigned to roles for a particular branch of the topic tree. An assignment applies to all topics with paths belonging to the branch unless overridden by a more specific assignment.

The *MODIFY_TOPIC* permission is required to add or remove a topic. The *UPDATE_TOPIC* permission is required to update a topic value.

The *READ_TOPIC* permission is required to subscribe to or fetch a topic. If a session does not have *READ_TOPIC* permission for a topic, the topic will be excluded from the results of subscription or fetch operations for the session. *READ_TOPIC* permissions are one factor the server's dynamic join of topic selectors to available topics. If a session's roles change – for example, perhaps a control session applies the *change roles* operation to the session – the server will reevaluate its topic selectors. The session will be subscribed to matching topics for which it now has permission and unsubscribed from the topics for which it no longer has permission.

The *SELECT_TOPIC* permission is required to use a topic selector, so controls the parts of the topic tree from which a session can subscribe or fetch. Given the *READ_TOPIC* permission controls access to topic paths, why is this useful? The answer is that some applications delegate subscription to a control session. A session that has *READ_TOPIC* permission but not *SELECT_TOPIC* permission for a

particular topic path cannot subscribe directly to topics belonging to the path. However, the session can be independently subscribed by a control session that has the *MODIFY_SESSION* global permission in addition to the appropriate *SELECT_TOPIC* permission.

Sometimes a topic is used to publish information to a single user, for a user to broadcast information, or to share data between a user's multiple sessions. In these cases, it can be unwieldy to set up lots of specialized topic permissions for the different security principals representing the users. An alternative is to create the topic as owned by a particular principal, using the *OWNER* topic property. A topic with the *OWNER* property grants full acccess to sessions authenticated with the named principal. Other sessions continue to be constrained by the configured topic permissions.

Premium features: persistence, replication, and fan-out

Three topic-related features are included in the separately licensed Scale and Availability pack.

Topic persistence logs a server's topic data to disk. Topic persistence allows a server to be stopped and restarted without needing to start a separate client to re-create topics and their values. It can provide faster time-to-recovery and is very useful during development when servers are frequently restarted or test data needs to be shared between developers and environments.

Topic replication mirrors the topic tree across a cluster of peer servers. This improves system availability – the topic data can survive the loss of individual servers – and provides a consistent view of the data to each client session regardless of the server that hosts the session.

Fan-out is designed for replication of topic data between different geographies. Fan-out links can be configured to mirror selected parts of the topic tree from a primary server or cluster of primary servers to one or more secondary servers. The secondary servers present a read-only view of the topic data; updates can only be made through the primary server. Some Diffusion systems use fan-out within a data center, to separate a primary data tier of servers from a secondary tier of servers that host customer sessions. This design allows the secondary tier to be scaled independently to support millions of sessions.

Introducing sessions

To interact with a Diffusion server, the first step a client application takes is to create a session.

Sessions

A Diffusion server can host tens of thousands of sessions. Each session holds information about an active client. The session information includes security details:

- the authenticated principal (typically a user name), and the set of roles granted to the session
- the topic selectors which the client has subscribed to
- the resolved subscriptions
- a set of session properties that describe various attributes of the session

Each session is assigned a unique session identity.

Creating a session

To create a session, a client opens an initial connection to the server. The session attributes set through the client API determine the transport protocol to use for the connection and the host name and port of the server. All Diffusion client libraries support the WebSocket transport protocol, with the option of Transport Layer Security (TLS, more commonly known as SSL). The Java and JavaScript client libraries also support an HTTP long polling (XHR) transport; again, this can be protected with TLS. The HTTP long polling transport can be configured as a fallback if a WebSocket connection cannot be established.

Once a connection has been established, the client sends the server the authentication details that have been set by the application. These may be empty if the application is connecting anonymously, or a principal name and credentials such as a password. The server assigns a session identity, and passes the connection and authentication details to the configured authentication handlers, including those registered by other sessions that have the `AUTHENTICATE` security permission. If the authentication details are accepted by one of the handlers, the server creates a session.

Disconnection and reconnection

Connections between clients and servers are long-lived, but a session can outlive individual connections. A client can become disconnected from the server, and reconnect to the server without loss of the session. Reconnection must occur within the reconnection timeout configured for the server. The default reconnection timeout is 5 minutes.

Diffusion reconnection is precise and lossless. If a client reconnects successfully, no data will have been lost and the only effect visible to the application is the delay while the connection is reestablished. The lossless behavior is achieved by storing the latest network messages sent to and from the client in recovery buffers. On reconnection, the client and server re-send the messages that each other has not received. In some cases, for example if there was a lot of in-flight data buffered in a network router, the recovery buffer size can be insufficient (the sizes can be tuned through configuration), and recovery is not possible. The server reliably detects this and closes the session.

While a session is disconnected, the server continues to queue topic updates for the session's subscriptions and will forward them to the session on reconnection. Queuing the updates requires server memory, and if the queue exceeds configurable limits the session will be closed. Conflation can be used to combine updates for the same topic, with the benefits of reducing the server memory required by the queue, the likelihood of the queue overflowing, and the work required to bring the session back up to speed.

Session properties

Each session has a set of properties that describe various attributes of the session. They provide a way to classify sessions into groups. There are two types of session property. Fixed session properties are assigned by the server and include the session id; the client library type; the network protocol and network address; if enabled on the server, location information derived from the network address; the name of the server the session is connected to; the security principal and roles; and more. User-defined session properties are arbitrary key/value pairs assigned by the application.

Session properties are closely integrated with Diffusion security. Only privileged sessions can modify session properties. This is either achieved directly using the `setSesssionProperties` operation or by registering an `Authenticator` implementation and participating in the authentication of sessions. A client may propose session properties when creating the session, but these are only suggestions that the authentication handler can modify or ignore entirely.

A session filter is an expression that selects sessions by their session properties. Many control operations in the API allow a session filter to be provided. For example, the `subscribeByFilter` operation allows a session with the `MODIFY_SESSION` security permission to subscribe all sessions satisfying a session filter to a set of topics.

A session with `REGISTER_HANDLER` and `VIEW_SESSION` permissions can register a session properties listener with the server. The server will notify the session when other sessions are opened, closed, or the session properties are changed.

Control clients

A control client is an application client that performs back-end tasks such as creating topics and updating topics or managing other sessions. The Diffusion client API provides many control operations. The wide range of features that allow a session to monitor and control other sessions

is particularly unusual. We have mentioned some of these above: authenticating sessions; setting session properties; monitoring session activity with session properties listeners; subscribing other sessions to topics.

The term "control client" describes the role of a client in the application system and distinguishes it from other clients running on web browsers, mobile devices, or IoT devices. Control clients authenticate to the server using accounts with elevated security permissions.

Sessions opened by control clients are referred to as control sessions.

Messaging and asynchronous application services

Messaging is an alternative way to pass information between sessions without using topics. One session sends a request message, and one or more other sessions reply with a response message. Unlike data published through topics, messages are transient and not stored on the server.

Messages are sent to a message path, which provides a delivery context. Message paths are arbitrary and can be chosen freely by the application. Just like topic paths, a message path is a string of parts separated by the `/` character. This segmented structure allows a recipient session to register a common handler for requests for paths that share a common prefix. Also like topic paths, security permissions assigned to a path are inherited by sub-paths unless a more specific permission assignment is found.

Each request and response message carries a single value. Values belong to one of the standard Diffusion data types. Applications should be written so that the sending and receiving sessions agree on the data types to use for a particular message path.

You can think of a message path as the name of an application service. For example, an application might implement a message handler that receives requests on the message path `account/23237` and returns details of the customer account `23237`. The request/reply facility provided by messaging shares similarities with REST services.

Three distinct messaging patterns are supported by Diffusion. Each routes request and response messages via the server. The three patterns are as follows.

- A session sends a request to the server, for a message path. If one or more control clients has registered a message handler for the message path, the server will select one of the handlers to respond to the request. If there are no registered handlers, the request will fail. The requesting session must have the `SEND_TO_MESSAGE_HANDLER` security permission for the message path. The control clients must have `REGISTER_HANDLER` permission. A control client that has `VIEW_SESSION` permission can additionally instruct the server to forward a subset of the sender's session properties with the request.
- A control client sends a request to a session, addressed by its session identity, for a message path. The session responds. The control client must have `SEND_TO_SESSION` permission for the message path.
- A control client sends a request to many sessions, addressed by a session filter, for a message path. Each session responds. The control client must have `SEND_TO_SESSION` permission for the message path.

After sending a request, the sender will receive either a response or a failure from each recipient.

Session replication

Session replication keeps redundant copies of session data within a cluster of peer servers. This improves system availability as a session can survive the loss of an individual server or failure of the network path to its current server.

If a session is connected to a server that belongs to a session replication enabled cluster, and then becomes disconnected, it will attempt to reconnect to the original server. A properly configured load balancer can detect that the original server is unavailable and re-route the reconnection request to

a second server in the cluster. The second server can recover session data and continue the session. This process is known as "fail over". Unlike with reconnection, in-flight messages can be lost during fail over, and the application will be unsubscribed and re-subscribed to topics.

What's new in Diffusion 6.7?

The latest version of Diffusion contains new features, performance enhancements and bug fixes.

A complete list of the latest updates and known issues can be found in the Release Notes available at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html.

Session trees

Diffusion's best-in-class data wrangling capabilities continue to grow with the addition of session trees.

A session tree is a custom view of the topic tree presented to an individual client session. Each session can be presented with a unique session tree based on its session properties.

This powerful new feature makes it even easier to tailor your solution to meet security, optimization and personalization requirements.

For more information, see Session trees on page 67.

CDC database adapter

You can now easily connect Diffusion to a wider range of data sources. The new Change Data Capture adapter enables easy, real-time streaming of data changes from relational databases to JSON topics within Diffusion.

The adapter uses Debezium to provide a wide range of configuration options. MySQL and PostgreSQL databases are tested and officially supported for production use.

For more information, see CDC database adapter on page 640.

Missing topic notifications from remote servers

Remote topic views are a powerful way to share data between remote servers. They were introduced in Diffusion 6.5, enabling your real-time application to overcome geographical distance and firewalls.

This release improves the support for missing topic notifications when using a remote topic view. Notifications are now propagated in the same way as the older fan-out system.

This means that remote topic views are now a full replacement for fan-out, which is deprecated in this release.

JSON patch clause for topic views

You can now use topic views to apply a JSON patch to the values within a JSON topic.

This enables you to use all the power of the JSON Patch standard to transform and wrangle topic data.

For more information, see Patch clause on page 58.

Other improvements

There are many more new features and improvements, including: a new topic view "type" clause enabling you to change the topic type of generated reference topics; dramatically improved performance when replicating time series topics; the ability to create and maintain metric collectors from the client APIs, not just the management console; and support in the C SDK for closing sessions by specifying a session ID.

Deprecations

Some older features are now deprecated:

- Routing topics (you should now use the more powerful session trees)
- Fan-out (you should now use remote topic views)
- Callbacks in the Java and .NET SDKs

See Upgrading from version 6.6 to version 6.7 on page 673 and the release notes at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html for full details.

Related concepts

Upgrading Guide on page 646

If you are planning to move from an earlier version of Diffusion, review the following information about changes between versions.

Part II

Quick Start Guide

Our Quick Start Guide explains how to set up Diffusion and start sending and receiving data using your language of choice.

Visit the Quick Start Guide to get up and running with Diffusion.

Other useful resources to help you get started:

- Download site: https://www.pushtechnology.com/developers/releases/6.7
- Support center: https://www.pushtechnology.com/support/
- Stack Overflow: http://stackoverflow.com/questions/tagged/push-diffusion
- Github: https://github.com/pushtechnology/

Part III

Design Guide

This guide describes the factors to consider when designing your Diffusion solution.

In this section:

- Support
- Designing your data model
- Designing your solution
- Security

Support

When designing your solution, refer to the support information to ensure compatibility between the Diffusion server and your hardware, software, and operating systems. This section also provides information about the capabilities of the Diffusion clients and the platforms the clients are supported on.

You can also refer to the Upgrading Guide to review the compatibility between different versions of Diffusion. For more information, see Interoperability.

System requirements for the Diffusion server

Review this information before installing the Diffusion server.

The Diffusion server is certified on the system specifications listed here. In addition, the Diffusion server is supported on a further range of systems.

Certification

Push Technology classes a system as certified if the Diffusion server is fully functionally tested on that system.

We recommend that you use certified hardware, virtual machines, operating systems, and other software when setting up your Diffusion servers.

Support

In addition, Push Technology supports other systems that have not been certified.

Other hardware and virtualized systems are supported, but the performance of these systems can vary.

More recent versions of software and operating systems than those we certify are supported.

However, Push Technology can agree to support Diffusion on other systems. For more information, contact Push Technology.

Physical system

The Diffusion server is certified on the following physical system specification:

- Intel™ Xeon™ E-Series Processors
- 8 Gb RAM
- 8 CPUs
- 10 Gigabit NIC

Network, CPU, and RAM (in decreasing order of importance) are the components that have the biggest impact on performance. High performance file system and disk are required. Intel hardware is used because of its ubiquity in the marketplace and proven reliability.

Virtualized system

The Diffusion server is certified on the following virtualized system specification:

Host

- Intel Xeon E-Series Processors
- 32 Gb RAM

- VMware vSphere® 5.5

Virtual machine

- 8 VCPUs
- 8 Gb RAM

When running on a virtualized system, over-committing VCPUs (assigning too many VCPUs compared to the processors available on the host) can cause increased latency and unpredictable performance. Consult the VMWare Performance Best Practices documentation for details.

Operating system

Diffusion is certified on the following operating systems:

- Red Hat® 7.2+
- Windows™ Server 2012 R2 and 2016

We recommend you install your Diffusion server on a Linux™-based operating system with enterprise-level support available, such as Red Hat Enterprise Linux.

Operating system configuration

If you install your Diffusion server on a Linux-based operating system and do SSL offloading of secure client connections at the Diffusion server, you must disable transparent huge pages.

If you install your Diffusion server on a Linux-based operating system but do not do SSL offloading of secure client connections at the Diffusion server, disabling transparent huge pages is still recommended.

Having transparent huge pages enabled on the system your Diffusion server runs on can cause extremely long pauses for garbage collection. For more information, see https://access.redhat.com/solutions/46111.

Java™

The Diffusion server is supported on any JVM that meets the following requirements:

- The JVM has either passed the Java TCK or is an official AdoptOpenJDK build.
- The JVM version is Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later). We recommend that you use the latest available minor release of the JVM, regularly review and update the JVM as new minor releases become available, and prefer a distribution that provides regular updates.
- The JVM is HotSpot™ based.
- The target platform is Linux, macOS®, or Windows.

There is a wide variety of free and commercial JDK distributions satisfying these requirements including:

- Oracle JDK (https://www.oracle.com/technetwork/java/javase/downloads/index.html). Production use requires a commercial licence from Oracle.
- AdoptOpenJDK (https://adoptopenjdk.net/).
- Amazon Corretto (https://aws.amazon.com/corretto/).
- Azul Zulu (https://www.azul.com/downloads/zulu/).
- Red Hat OpenJDK (https://developers.redhat.com/products/openjdk/overview/).

We recommend against using Oracle OpenJDK (http://jdk.java.net/). Oracle OpenJDK is a reference implementation, and release updates are provided for a limited period, typically only six months.

Diffusion 6.7 is certified against the following JVMs:

- AdoptOpenJDK 8u212

- AdoptOpenJDK 11.0.3+7
- Oracle JDK 8u131

Reduced footprint JVM distributions

Some distributions have optional packages that are minimal versions of the JVM and omit certain components, either to reduce the size of the JVM (and so the required disk space and time to download), or as part of security hardening. Additionally, the end user may remove unnecessary components from a JDK. JDKs up to version 8 had a standard reduced footprint repackaging called the "Java Runtime Environment (JRE)", but this name has now been dropped to allow distributions or the end user to choose from a variety of different repackaging strategies.

Examples of components that are frequently removed include:

- Tools that are only required by developers (e.g. the javac compiler; Java Mission Control; jconsole). This was the primary focus of the JRE.
- Graphical user interface libraries. Packages without these components are typically referred to as "headless".

Diffusion does not require developer tools nor graphical user interface libraries. Although Push Technology Ltd supports the use of Diffusion with a reduced footprint JVM, we recommend that a full JDK installation is used. This is because Java's diagnostic tools, in particular Java Mission Control, have proven to be particularly useful as part of performance tuning and problem diagnosis.

JVM configuration

If you do SSL offloading of secure client connections at the Diffusion server, you must ensure that you constrain the maximum heap size and the maximum direct memory size so that together these to values do not use more than 80% of your system's RAM.

Networking

Push Technology recommends the following network configurations:

- 10 Gigabit network
- Load balancers with SSL offloading
- In virtualized environments, enable SR-IOV.

  For more information about how to enable SR-IOV, see the documentation provided by your virtual server provider. SR-IOV might be packaged using a vendor-specific name.

Validation script

You can run an environment validation script to detect problems with the install environment.

The script uses the jjs command line tool which is included with Java 8 and above.

1. In the directory where you installed Diffusion, navigate to `/bin`.
2. Run the environmentValidation.js script:
   - On Linux, run `./environmentValidation.js`
   - On Windows, make sure `<java_home>` is added to your path, then run: `jjs environmentValidation.js`

Note that macOS is not certified as a production system for Diffusion, so the validation script does not support macOS. You can still install and run Diffusion on macOS for development purposes.

Client requirements

For information about the supported client platforms, see Platform support for the Diffusion API libraries on page 30.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Platform support for the Diffusion API libraries

Review this information when designing your clients to determine what platforms and transports the Diffusion client libraries are supported on.

Supported platforms and protocols for the client libraries

Table 1: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
JavaScript®	es6 (TypeScript 1.8)	WebSocket HTTP (Polling XHR)
Apple® for iOS®	**Development environment** Xcode 12.4 (iOS 10.0 SDK) **Runtime support**	WebSocket

Platform	Minimum supported versions	Supported transport protocols
	Device architectures: armv7, armv7s, arm64 Simulator architectures: i386, x86_64	
Apple for OS X®/macOS	**Development environment** Xcode 12.4 (OS X/macOS 10.12 SDK) **Runtime support** Device architectures: x86_64	WebSocket
Apple for tvOS™	**Development environment** Xcode 12.4 **Runtime support** Device architectures: arm64 Simulator architectures: x86_64	WebSocket
Android™	API 19 / v4.4 / KitKat and later **Note:** Push Technology provides only best-effort support for Jelly Bean (API 16-18, v4.1-4.3).	WebSocket HTTP (polling)
Java	Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later) **Note:** The JVM must be HotSpot based. **Note:** The JVM must either have passed the Java TCK or be an official AdoptOpenJDK build.	WebSocket HTTP (Polling)

Platform	Minimum supported versions	Supported transport protocols
.NET	Microsoft® .NET Standard 2.0	WebSocket
C for Linux	Red Hat and CentOS™, version 7.2 and later Ensure that you use a C99-capable compiler.	WebSocket
C for Windows	Visual C Compiler 2013 or later, Windows 7 or later	WebSocket
C for OS X/macOS	For building using GCC, use Xcode 8.0 or later	WebSocket
Python	CPython 3.7.8 or later, or CPython 3.8.5 or later	

Note: Protocols are supported for both secure and standard connections.

Feature support in the Diffusion API

Review this information when designing your clients to determine which APIs provide the functionality you require.

Features are sets of capabilities provided by the Diffusion API. Some features are not supported or not fully supported in some APIs.

The Diffusion libraries also provide capabilities that are not exposed through their APIs. Some of these capabilities can be configured.

Note: The Python client currently implements limited capabilities. See the Python API documentation for details.

Table 2: Capabilities provided by the Diffusion client libraries

Capability	JavaScript	Apple	Android	Java	.NET	C
Connecting to the Diffusion server on page 186						
Connecting on page 188	✓	✓	✓	✓	✓	✓
Connecting with multiple transports on page 189	✓	✗	✓	✓	✗	✗
Asynchronous connections on page 190	✓	✓	✓	✓	✓	✓
Synchronous connections on page 191	✗	✗	✓	✓	✓	✓
Connecting on page 188	✗	✓	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Connecting on page 188	✓	✗	✓	✓	✗	✓
Connect securely on page 194	✓	✓	✓	✓	✓	✓
Configure the SSL context or behavior on page 194	✓	✓	✓	✓	✓	✗
Connecting through an HTTP proxy on page 199	✗	✓	✓	✓	✓	✗
Connecting through a load balancer on page 202	✓	✓	✓	✓	✓	✓
Connecting through a load balancer on page 202	✓	✓	✓	✓	✗	✓
Reconnect to the Diffusion server on page 203						
Reconnect to the Diffusion server on page 203	✓	✓	✓	✓	✓	✓
Configuring reconnection on the client on page 203	✓	✓	✓	✓	✓	✓
Specifying a reconnection strategy on page 206	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Configuring the recovery buffer on the client on page 205	✗	✓	✓	✓	✓	✓
Monitoring the connection activity on page 206	✓	✓	✓	✓	✓	✓
Using TCP state on page 206	✓	✓	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Pinging the Diffusion server on page 218	✓	✓	✓	✓	✓	✓
Change the security principal and credentials associated with your client session on page 219	✓	✓	✓	✓	✓	✓
Receiving data from topics						
Subscribe to a topic or set of topics	✓	✓	✓	✓	✓	✓
Using streams for subscription on page 227	✓	✓	✓	✓	✓	✓
Using streams for subscription on page 227	✓	✓	✓	✓	✓	✓
Fetching the current value of a topic on page 233	✓	✓	✓	✓	✓	✓
Managing topics						
Managing topics on page 244	✓	✓	✓	✓	✓	✓
Slave topics (DEPRECATED) on page 87	✓	✓	✓	✓	✓	✓
Using time series topics on page 297	✓	✓	✓	✓	✓	✓
Create a topic and set initial value	✓	✓	✓	✓	✓	✓
Adding topics with topic specifications on page 245	✓	✓	✓	✓	✓	✓
Adding topics with topic specifications on page 245	✓	✓	✓	✓	✓	✓
Create a topic with metadata	✓	✓	✓	✓	✓	✓
Listen for topic events (including topic has subscribers and topic has zero subscribers)	✗	✓	✓	✓	✓	✗

Capability	JavaScript	Apple	Android	Java	.NET	C
Receiving topic notifications on page 240	✓	✓	✓	✓	✓	✓
Delete a topic	✓	✓	✓	✓	✓	✓
Delete a branch of the topic tree	✓	✓	✓	✓	✓	✓
Automatic topic removal on page 122	✓	✓	✓	✓	✓	✓
Updating topics						
Updating topics on page 291	✓	✓	✓	✓	✓	✓
Perform exclusive updates	✓	✓	✓	✓	✓	✓
Perform non-exclusive updates	✓	✓	✓	✓	✓	✗
Managing subscriptions						
Managing subscriptions on page 309	✓	✓	✓	✓	✓	✓
Managing subscriptions on page 309	✓	✓	✓	✓	✓	✗
Managing subscriptions on page 309	✗	✗	✓	✓	✓	✗
Handling subscriptions to missing topics on page 254	✓	✓	✓	✓	✓	✓
Using request-response messaging on page 323						
Sending request messages to a message path on page 325	✓	✓	✓	✓	✓	✓
Sending request messages to a session on page 331	✓	✓	✓	✓	✓	✓
Sending request messages to a session filter on page 338	✓	✓	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Sending request messages to a session on page 331	✓	✓	✓	✓	✓	✓
Sending request messages to a message path on page 325	✓	✓	✓	✓	✓	✓
Managing security						
Authenticating new sessions on page 342	✓	✗	✓	✓	✓	✓
Updating the system authentication store on page 356	✓	✗	✓	✓	✓	✓
Updating the security store on page 367	✓	✗	✓	✓	✓	✓
Updating the security store on page 367	✓	✗	✓	✓	✓	✓
Topic ownership on page 156	✓	✓	✓	✓	✓	✓
Managing other clients						
Receiving notifications of client session events and their session properties on page 378	✓	✗	✓	✓	✓	✓
Getting properties of specific client sessions on page 382	✓	✗	✓	✓	✓	✓
Update the properties of specific client sessions on page 383	✓	✗	✓	✓	✓	✓
Receiving notifications of client queue events on page 383	✗	✗	✓	✓	✓	✗
Conflate and throttle clients	✗	✓	✓	✓	✓	✗

Capability	JavaScript	Apple	Android	Java	.NET	C
Closing client sessions on page 377	✗	✓	✓	✓	✓	✗
Other capabilities						
Flow control on page 384	✗	✗	✓	✓	✓	✓

License types

Some advanced Diffusion features are limited to certain license types.

Licenses can also limit the number of concurrent sessions or topics.

You can obtain a free Community Production or Evaluation license at community.pushtechnology.com.

The Community Evaluation license is time-limited and intended to help you explore all Diffusion features.

The Community Production license is suitable for production of small-scale applications hosted on a single server.

If you need to buy or upgrade a paid commercial license, contact sales@pushtechnology.com . The sales team can also provide a more capable Evaluation license if required.

Table 3: License capabilities

Feature	Restricted Default	Community Production	Community Evaluation	Commercia	Commercia with Scale & Availability Pack
Core Diffusion features	✓	✓	✓	✓	✓
Production use allowed?		✓		✓	✓
Maximum sessions	5	1000	25	Varies	Varies
Maximum topics	1000	1000	Unlimited	Varies	Varies
Maximum CPU cores	8	8	8	Varies	Varies
Persistence			✓		✓
Remote connections (remote topic views and fan-out)			✓		✓
Session replication			✓		✓
Topic replication			✓		✓

Feature	Restricted Default	Community Production	Community Evaluation	Commercia	Commercia with Scale & Availability Pack
Cluster management (including Prometheus)			✓		✓

Commercial license restrictions

A specific commercial license may have restrictions such as an expiry date, allowed IP range or a maximum number of concurrent sessions, depending on your agreement with Push Technology. See License restrictions on page 408 for details of possible restrictions.

Protocol support

Each client supports varying transports. A table of the supported transports for each client is presented here.

All protocols supported by Diffusion can be used for both secure (using TLS) and standard connections. For more information, see SSL and TLS support on page 38.

The following table lists the protocols supported for each client:

Table 4: Supported protocols by client

Client	WebSocket	HTTP Polling
JavaScript API	✓	✓
Apple API	✓	
Android API	✓	✓
Java API	✓	✓
.NET API	✓	
C API	✓	

The JavaScript client is fully supported only on certain browsers. Best effort support is provided for other browsers but the software/hardware combination might not be reproducible, particularly for mobile browsers. For more information about supported browsers, see Browser support on page 39.

SSL and TLS support

Diffusion supports only those SSL versions and cipher suites with no known vulnerabilities.

The following SSL and TLS versions are supported by default:

- TLSv1

- TLSv1.1
- TLSv1.2
- TLSv1.3 (Java 11 JDK recommended)

The following cipher suites are supported by default:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA

For more information, see Network security on page 487.

Browser support

Some of the client libraries are intended to be run within browser environments. Diffusion clients can use most commercial browsers and their variants. However, some Diffusion API protocols might not be available.

Diffusion supports the latest release of the following browser versions based on the original Diffusion release date.

Table 5: Supported browsers

Browser	Version
Google Chrome™	53
Mozilla Firefox®	49
Microsoft Internet Explorer®	11
Apple Safari®	8

- Push Technology runs full regression tests on the browsers and versions documented above for every patch release.
- Push Technology carries out basic validation testing on the latest versions of these browsers but full support is available only at the next minor release.
- Support for older versions of browsers is provided on a best-effort basis, unless specified otherwise.
- Support for other browsers is provided on a best-effort basis.

Mobile browsers

We do not test our JavaScript client libraries with mobile browsers or within mobile applications that wrap a browser application in native code. If you are developing a Diffusion client for a mobile platform, such as iOS or Android, we recommend that you use the provided client libraries for these platforms to develop a native application.

Diffusion JavaScript clients running within a native wrapper or in a mobile browser are supported on a best effort basis and we might not be able to provide support for older versions of the mobile platform.

Browser limitations

Some browsers cannot use all the Diffusion protocols or features. If you experience problems when developing with protocols or client libraries that use the browser, check whether the browser supports this function.

Browser environments are not uniform and some browsers might have functional limitations. It is important to be aware of these limitations when developing for compliance with target browsers.

WebSocket limitations

WebSocket is an Internet Engineering Task Force (IETF) protocol used by Diffusion to provide a full-duplex communication channel over a single TCP connection. It is not supported by all browser versions.

Table 6: Support for WebSocket

Version	WebSocket support?
Internet Explorer 9.0 and earlier	NO
Internet Explorer 10.0 and later	YES (see note)
Firefox	YES
Chrome	YES
Safari	YES
Opera®	YES
iOS	YES
Android	YES

Note: Internet Explorer 11 contains a bug that causes WebSocket connections to be dropped after 30 seconds of inactivity. To work around this problem set the system ping frequency to 25 seconds or less. You can set the system ping frequency in the `Server.xml` configuration file.

Cross-origin resource sharing limitations

CORS allows resources to be accessed by a web page from a different domain. Some browsers do not support this capability.

Table 7: Support for CORS

Version	WebSocket support?
Internet Explorer 9.0 and earlier	NO
Internet Explorer 10.0 and later	YES
Firefox	YES
Chrome	YES
Safari	YES

Version	WebSocket support?
Opera	YES
iOS	YES
Android	YES

Browser connection limitations

Browsers limit the number of HTTP connections with the same domain name. This restriction is defined in the HTTP specification (RFC2616). Most modern browsers allow six connections per domain. Most older browsers allow only two connections per domain.

The HTTP 1.1 protocol states that single-user clients should not maintain more than two connections with any server or proxy. This is the reason for browser limits. For more information, see RFC 2616 – Hypertext Transfer Protocol, section 8 – Connections.

Modern browsers are less restrictive than this, allowing a larger number of connections. The RFC does not specify how to prevent the limit being exceeded. Either connections can be blocked from opening or existing connections can be closed.

Table 8: Maximum supported connections

Version	Maximum connections
Internet Explorer 7.0	2
Internet Explorer 8.0 and 9.0	6
Internet Explorer 10.0	8
Internet Explorer 11.0	13
Firefox	6
Chrome	6
Safari	6
Opera	6
iOS	6
Android	6

Some Diffusion protocols like HTTP Polling (XHR) use up to two simultaneous connections per Diffusion client. It is important to understand that the maximum number of connections is per browser and not per browser tab. Attempting to run multiple clients within the same browser might cause this limit to be reached.

Reconnection can be used to maintain a larger number of clients than is usually allowed. When TCP connections for HTTP requests are closed, the Diffusion sends another HTTP request which the server accepts. Be aware of cases where Diffusion tries to write a response to closed polling connections before the client can re-establish them. This behavior results in an IO Exception and the Diffusion server closes the client unless reconnection is enabled. When the client tries to re-establish the poll, it is aborted.

Another way to get around browser limits is by providing multiple subdomains. Each subdomain is allowed the maximum number of connections. By using numbered subdomains, a client can pick a random subdomain to connect to. Where the DNS server allows subdomains matching a pattern to be resolved as the same server, tab limits can be mitigated substantially.

Designing your data model

Distribute your data in a data model that fits the needs of your organization and customers.

There are a number of things to consider when designing your data model:

- The structure of your topic tree
- The types of topic to use
- The format of your data
- How you publish data to topics
- Your conflation strategy
- Whether you also use messaging to send data.

These considerations are not separate. The decisions you make about one aspect of your data model affect other aspects.

The data model is defined on the Diffusion server by your publishers or clients. The topic structure, topic types, and data format only persist on the Diffusion server through a restart or upgrade if the topic persistence feature is enabled.

Design your solution to create your data model on the Diffusion server afresh after the Diffusion server is restarted or upgraded.

Topic tree

Diffusion primarily distributes data using a pub-sub model, where content is published to topics. These topics are arranged as a tree.

What is the topic tree?

The topic tree is the organizational structure of the Diffusion topics. A topic of any type can be created any point in the topic tree where a topic does not already exist.

Locations in the topic tree are referred to by their topic path, which is the level names in the tree that lead to the topic, separated by the slash character (/).

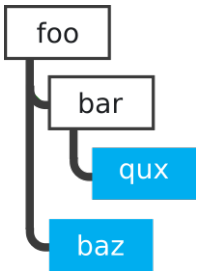


Figure 1: Example topic tree

In the preceding image, topics exist at baz and qux. The topic path for baz is foo/baz. The topic path for qux is foo/bar/qux

You can create a topic at foo/bar/qux without having to create topics at foo or foo/bar beforehand.

There can be multiple topics that have the same name, but topic paths are unique.

When interacting with topics in the topic tree, your clients can reference a topic by its topic path or by a *topic selector* with a pattern expression that matches the topic path. Clients can use topic selectors to reference sets of topics in the tree that all match the topic selector's pattern expression.

Considerations when designing your topic tree

- Does the source information have a logical organization?

  If the data structure of the source information is complex, it can be mapped to a hierarchical structure.
- How many topics?

  If the number of topics is small, a flat topic tree might be appropriate.
- How do clients subscribe to the data?

  If there are topics that clients generally subscribe to together, these topics can be organized in the same branch of the topic tree. This enables a client use a topic selector to subscribe to the branch of the topic tree and receive updates for all topics in that branch.
- The size of your topic tree can be constrained by your hardware.

  An extremely large topic tree can cause long GC pauses. Ensure that you do sufficient testing with your topic tree before going to production.

  If the size of your topic tree structure is caused by a lot of duplication, use routing topics to reduce it.
- A topic cannot be bound to the / topic path. This is because each segment of a topic path must have one or more characters. This means there can be no single topic that acts as the root topic for all possible topics in the topic tree. Instead each top-level topic whose path contains a single part acts as the root topic for their branch of the topic tree.

  However, the / path can be used as a routing path when sending or receiving messages, which uses paths but does not use any topics that are bound to them.

Related concepts

Topic selectors on page 44
A topic selector defines a set of topics paths that identify topics. You can create a topic selector from a topic selector expression.

Topic views on page 50
A topic view is a dynamic way to map one part of a server's topic tree to another. This enables the server to transform topic data before sending it to clients. Remote topic views enable mapping of topics between separate servers.

Topic naming

Consider the following restrictions when deciding on your topic paths.

Restricted characters

Topics are identified by a path. A path is made up of path segments separated by the slash character (/).

Each path segment can be made up of one or more Unicode characters but must not contain any of the restricted characters mentioned below. The slash character (/) is not permitted in any path segment.

Reserved spaces

Paths starting with the path segment Diffusion are reserved for internal use.

Recommendations

Although all Unicode characters except the slash character (/) are supported, we recommend avoiding common regular expression metacharacters such as '*' and '?' in path names to avoid having to quote these characters in topic selector expressions.

Topic selectors

A topic selector defines a set of topics paths that identify topics. You can create a topic selector from a topic selector expression.

Topic selector expressions

Use topic selector expressions to create a topic selector of one of the types described in the following table. The type of the topic selector is indicated by the first character of the expression.

Table 9: Types of topic selector

Topic selector type	Initial character	Description
Path	>	A path expression must contain a valid topic path. A valid topic path comprises path segments separated by path separators (/). A path segment comprises one or more UTF characters except for slash (/). A path selector returns only the topic with the given path. See Path expression examples on page 45 If the first character is not one of #, ?, >, *, $, %, & or <, the initial '>' can be omitted.
Split-path	?	A split-path pattern expression contains a list of regular expressions separated by the / character. Each regular expression describes a part of the topic path. The selector returns topics for which each regular expression matches the part of the topic path at the corresponding level. See Split-path pattern expression examples on page 46
Full-path	*	A full-path pattern expression contains a regular expression. A full-path pattern topic selector returns topics for which the regular expression matches the full topic path. See Full-path pattern expression examples on page 47
Selector set	#	A selector set expression contains a list of selectors separated by the separator ////. A selector set topic selector returns topics that match any of the selectors. **Note:** Use the `anyOf()` method for a simpler method of constructing selector set topic selectors. See Selector set expression examples on page 48

Topic path prefixes

The topic selector capabilities in the Diffusion API provide methods that enable you to get the topic path prefix from a topic selector.

A topic path prefix is a concrete topic path to the most specific part of the topic tree that contains all topics that the selector can specify. For example, for the topic selector `?foo/bar/baz/.*/bing`, the topic path prefix is `foo/bar/baz`.

The topic path prefix of a selector set is the topic path prefix that is common to all topic selectors in the selector set.

Regular expressions in pattern expressions

Split-path pattern and full-path pattern expressions can contain any regular expression, with the following restrictions:

- A regular expression cannot be empty
- In split-path pattern expressions, a regular expression cannot contain the path separator (/)
- In full-path pattern expressions, a regular expression cannot contain the selector set separator (////)

Depending on what the topic selector is used for, regular expressions in topic selectors can be evaluated on the client or on the Diffusion server. For more information, see Regular expressions on page 49.

> **Note:** Regular expressions have a moderate cost in terms of CPU usage. More complex regular expressions, using features such as back tracking, have a higher cost. However, the principal performance determinant is the number of topic paths a pattern expression is applied to. Topic selector expressions are evaluated hierarchically against the topic tree. Expressions that have longer topic path prefixes select less of the tree and are less costly. Similarly, split path patterns can be used to rapidly hone in on the interesting sub-set of the topic tree, so are usually preferable to full path patterns. If you are using very general pattern expressions or complex regular expressions, be sure to test the performance impact under realistic conditions.

Descendant pattern qualifiers in pattern expressions

You can modify split-path or full-path pattern expressions by appending a descendant pattern qualifier. These are described in the following table:

Table 10: Descendant pattern qualifiers

Qualifier	Behavior
None	Select only those topics that match the selector.
/	Select only the descendants of the matching topics and exclude the matching topics.
//	Select both the matching topics and their descendants.

Path expression examples

The following table contains examples of path expressions:

Expression	Matches alpha/beta?	Matches alpha/beta/gamma?	Notes
>alpha/beta	yes	no	
alpha/beta	yes	no	

Expression	Matches alpha/beta?	Matches alpha/beta/gamma?	Notes
>/alpha/beta/	yes	no	This path expression is equivalent to the path expression in the preceding row. In an absolute topic path, single leading or trailing slash characters (/) are removed because the topic path is converted to canonical form. A path expression can match a maximum of one topic. The trailing slash in this example is not treated as a descendant qualifier and is removed.
>alpha/beta/gamma	no	yes	
alpha/beta/gamma	no	yes	
>beta	no	no	The full topic path must be specified for a path expression to match a topic.
>.*/.*	no	no	The period (.) and asterisk (*) characters are valid in path segments. In a path expression these characters match themselves and are not evaluated as part of a regular expression.
.*/.*	no	no	
>$topic$	no	no	This expression matches a single topic path `$topic`. The leading > is required because the first character is $.

Split-path pattern expression examples

The following table contains examples of split-path pattern expressions:

Expression	Matches alpha/beta?	Matches alpha/beta/gamma?	Notes
?alpha/beta	yes	no	
?alpha/beta/	no	yes	The trailing slash character (/) is treated as a descendant pattern qualifier in split-path pattern expressions. It returns descendants of the matching topics, but not the matching topics themselves.
?alpha/beta//	yes	yes	Two trailing slash characters (//) is treated as a descendant pattern qualifier in split-path pattern expressions. It returns matching topics and their descendants.
?alpha/beta/gamma	no	yes	
?beta	no	no	
?.*	no	no	Each level of a topic path must have a regular expression specified for it for a split-path pattern expression to match a topic.
?.*/.*	yes	no	
?alpha/.*//	yes	yes	In this pattern expression, “alpha/.*” matches all topics in the alpha branch of the topic tree. The descendant pattern qualifier (//) indicates that the matching topics and their descendants are to be returned.

Full-path pattern expression examples

The following table contains examples of full-path pattern expressions:

Expression	Matches alpha/beta?	Matches alpha/beta/gamma?	Notes
*alpha/beta	yes	no	
*alpha/beta/gamma	no	yes	
*alpha/beta/	no	yes	The trailing slash character (/) is treated as a descendant pattern qualifier in full-path pattern expressions. It returns descendants of the matching topics, but not the matching topics themselves.

Expression	Matches alpha/beta?	Matches alpha/beta/gamma?	Notes
*alpha/beta//	yes	yes	Two trailing slash characters (//) is treated as a descendant pattern qualifier in full-path pattern expressions. It returns matching topics and their descendants.
*beta	no	no	In a full-path pattern selector the regular expression must match the full topic path for a topic to be matched.
*.*beta	yes	no	The regular expression matches the whole topic path including topic separators (/).

Selector set expression examples

Use the `anyOf` methods to create a selector set `TopicSelector` object.

The following example code shows how to use the `anyOf(TopicSelector... selectors)` method to create a selector set topic selector:

```
// Use your session to create a TopicSelectors instance
TopicSelectors selectors = Diffusion.topicSelectors();

// Create topic selectors for the individual topic selector
 expressions
TopicSelector pathSelector = selectors.parse(">foo/bar");
TopicSelector splitPathSelector = selectors.parse("?f.*/bar\d+");
TopicSelector fullPathSelector = selectors.parse("*f.*\d+");

// Use the individual topic selectors to create the selector set
 topic selector
TopicSelector selector = selectors.anyOf(pathSelector,
 splitPathSelector, fullPathSelector);

// Use the topic selector as a parameter to methods that perform
 actions on topics or groups of topics
```

The following example code shows how to use the `anyOf(String... selectors)` method to create the same topic selector as in the previous code example, but in fewer steps:

```
// Use your session to create a TopicSelectors instance
TopicSelectors selectors = Diffusion.topicSelectors();

// Create the selector set topic selector by passing in a list of
// pattern expressions
TopicSelector selector = selectors.anyOf(">foo/bar", "?f.*/bar\d+",
 "*f.*\d+");

// Use the topic selector as a parameter to methods that perform
 actions on topics or groups of topics
```

Related concepts

Topic tree on page 42

Diffusion primarily distributes data using a pub-sub model, where content is published to topics. These topics are arranged as a tree.

Topic views on page 50
A topic view is a dynamic way to map one part of a server's topic tree to another. This enables the server to transform topic data before sending it to clients. Remote topic views enable mapping of topics between separate servers.

Regular expressions

Depending on what the topic selector is used for, regular expressions in topic selectors can be evaluated on the client or on the Diffusion server. On the Diffusion server, regular expressions are evaluated as Java-style regular expressions. On clients, regular expressions are evaluated according to the conventions of the client library.

The following client uses of topic selectors are evaluated on the Diffusion server:

- Subscribing to topics
- Unsubscribing from topics
- Subscribing another client to topics
- Unsubscribing another client from topics
- Fetching topic states
- Removing topics

The following client uses of topic selectors are evaluated on the client:

- Creating a stream to receive updates published to topics
- Creating a stream to receive messages sent on message paths

The regular expression evaluation on each of the client libraries and on the Diffusion server are all based on the same style of regular expressions. The behavior of topic selectors on the clients and on the Diffusion server is the same for all standard uses of regular expressions. More advanced or complex regular expressions might differ slightly in behavior.

See the following sections for platform-specific information.

On the Diffusion server

Topic selectors evaluated on the Diffusion server are evaluated as Java-style regular expressions and are based on `java.util.regex.Pattern`.

For more information about Java-style regular expressions, see Java regular expressions.

On Java and Android clients

Topic selectors evaluated on these clients are evaluated as Java-style regular expressions. There are no differences between how a regular expression is evaluated on these clients and how it is evaluated on the Diffusion server.

For more information about Java-style regular expressions, see Java regular expressions.

On .NET clients

Topic selectors evaluated on the .NET client are evaluated as .NET Framework regular expressions, these are compatible with Perl 5 regular expressions. For more information about .NET regular expressions, see https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx.

The .NET evaluation of regular expressions can differ from the Java evaluation of the same regular expression on the Diffusion server. For examples of how these can differ, see http://stackoverflow.com/a/545348.

Ensure that you test the behavior of complex regular expressions that you use with the .NET client.

On Apple clients

Topic selectors evaluated on the Apple client are evaluated according to the `NSRegularExpression` class, which uses ICU regular expression syntax. For more information about Apple regular expressions, see:

- https://developer.apple.com/library/ios/documentation/Foundation/Reference/NSRegularExpression_Class/
- http://userguide.icu-project.org/strings/regexp

.

The Apple evaluation of regular expressions can differ from the Java evaluation of the same regular expression on the Diffusion server. Ensure that you test the behavior of complex regular expressions that you use with the Apple client.

On JavaScript clients

Topic selectors evaluated on the JavaScript client are based on the ECMAScript standard. For more information about JavaScript regular expressions, see:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
- http://www.ecma-international.org/ecma-262/5.1/#sec-7.8.5

The JavaScript evaluation of regular expressions can differ from the Java evaluation of the same regular expression on the Diffusion server. Ensure that you test the behavior of complex regular expressions that you use with the JavaScript client.

On C clients

Topic selectors evaluated on the C client use PCRE. For more information about C regular expressions, see http://www.pcre.org/.

The C evaluation of regular expressions can differ from the Java evaluation of the same regular expression on the Diffusion server. Ensure that you test the behavior of complex regular expressions that you use with the C client.

Topic views

A topic view is a dynamic way to map one part of a server's topic tree to another. This enables the server to transform topic data before sending it to clients. Remote topic views enable mapping of topics between separate servers.

Source topics and reference topics

A topic view maps topics from one part of the topic tree to another. It takes a set of existing source topics and dynamically creates a set of reference topics, based on a declarative topic view specification.

The reference topics can be a simple mirror, but a topic view can transform the topics in various ways, for example by changing their values or properties, or throttling the rate of publication.

You can use a remote topic view to map part of the topic tree on one Diffusion the Diffusion server to another.

You can also use topic view inserts to merge data from topics other than the source topics into the reference topics.

Topic views are useful to create reference topics with the exact data required by clients, minimising the need to process data on the client. For example, you can transform a source topic with a large, complex JSON value into a set of simpler reference topics. Clients can then subscribe only to the individual topics they need.

Reference topics are read-only. They cannot be updated, nor can they be created or removed directly. Otherwise, they behave just like standard topics. Each reference topic has the same topic type as its source topic.

Client sessions can subscribe to a reference topic, and can fetch its current value if it has one.

The topic specification of a reference topic is derived from the topic specification of the source topics. A reference topic has the same topic type as its source topic.

The source topics of a topic view are defined by a topic selector.

If a source topic is removed, reference topics that are derived from it will automatically be removed. If a topic is added that matches the source topic selector of a topic view, a corresponding reference topic will be created. Removing a topic view will remove all of its reference topics.

There is no hard-coded limit to the number of topic views you can create.

One-to-one mapping and expansion

In most topic views, the mapping is one-to-one: each source topic generates a single reference topic.

Starting in Diffusion 6.4, a topic view can use an expand value directive to create multiple reference topics from the values within a JSON source topic.

This is useful if you have a complex JSON topic with more information than each client requires. You can use an expand value directive to unpack the topic into multiple reference topics, and each client can subscribe to the relevant topics.

> **Note:** Topic view expansion is a powerful feature which should be used with care to avoid performance issues. If the topic structure of the generated reference topics changes frequently, it can lead to high server load.

See below for full details of how to use an expand value directive.

Topic view specifications

The following is a simple topic view specification that mirrors all topics below the path a to reference topics below the path b.

```
map ?a// to b/<path(1)>
```

A topic view with this specification will map a source topic at the path a/x/y/z to a reference topic at the path b/x/y/z. The specification is simple, so the reference topic will exactly mirror the source topic.

A general topic view specification contains these clauses:

- The **source topic clause** identifies the source topics.
- The **path mapping clause** determines how reference topic paths are derived from the source topic paths.
- The **insert clause** (optional) can be used to specify values or partial values from topics that are not part of the specified source topics; the specified data is inserted into reference topics (must be JSON).
- The **patch clause** (optional) can be used to apply a change to the value of the reference topic with respect to the source topic, as specified using the JSON Patch format.
- The **topic property mapping clause** determines how reference topic properties are derived from source topic properties.

- The **topic value mapping clause** (optional) determines how reference topic values are derived from source topic values.
- The **throttle clause** (optional) constrains the rate at which each reference topic is updated when its source topic is updated.
- The **preserve topics clause** (optional) affects the removal of reference topics that are derived from source topic values, giving you the option to preserve the reference topic even after the source topic value changes.
- The **separator clause** (optional) enables you to customize how topic values containing a `/` character are interpreted when creating a reference topic path.
- The **type** clause (optional) enables you to specify that the created reference topic has a different topic type than the source topic.

For readability, clauses can be separated by line endings. Lines prefaced with # are treated as comments.

An example topic view specification is:

```
map ?A//
from Server1
to <path(1)>
as <value(/foo)>
# Join 2 topics
insert Topic2 at /T2
insert Topic3 at /T3
throttle to 1 update every minute
```

Source topic clause

The source topic clause begins with the map keyword and is followed by a topic selector.

The following is an example of a source topic clause:

```
map ?foo/bar//
```

This matches the topics "foo", "bar" and all of their descendants. See Topic selectors on page 44 for the full topic selector syntax, which enables you to specify part of the topic tree using powerful regular expressions.

> **Note:** If any of the topic names contain spaces, wrap the selector in quotation marks. For example, instead of `map ?Results/Home games//`, use `map "?Results/Home games//"`.

When evaluating a topic view, routing topics are ignored even if they match the selector.

Reference topics are valid source topics. Chaining of topic views is supported; that is, a reference topic created by one topic view can be the source topic of another topic view.

Path mapping clause

The path of a reference topic is derived from the source topic according to the topic view path mapping. Both the path of the source topic and its value can be used to determine the path of the reference topic.

A path mapping clause begins with the `to` keyword and is followed by a **path mapping template**.

A path mapping template is a topic path with embedded directives. Directives are evaluated when creating the topic reference and substituted into the topic path.

Directives are delimited by angle brackets (<, >) and consist of the name of the directive and a list of parameters. The parameter list is comma-separated and surrounded by parentheses ((,)).

The following example adds a simple path mapping clause to the previous source topic clause:

```
map ?foo/bar// to /foo/bar/All/<path(3)>
```

The path mapping clause is the part after the `to` keyword. `path` is a directive, with "3" as its only parameter. The clause will make reference topics for all the topics under `foo/bar` at `foo/bar/All` in the topic tree.

There are three types of path mapping directive:

Source path directives

A source path directive matches part of the source path as defined by the source topic clause. This enables you to derive the path of the reference topics from all or part of the path of the source topics.

The syntax is `<path(start, number)>`:

- `start` is the index of part of the source path (where the top level of the topic tree is indexed as 0)
- `number` is the number of parts to include (optional; if not specified, the selection will extend to the end of the source path)

For example, if you have a source path `a/b/c/d`:

- The source path directive `<path(0, 2)>` is mapped to the reference topic path `a/b`. The `start` parameter of 0 means start at index 0, which is the topic `a`, and the `number` parameter of 2 means to take two parts of the tree in total.
- The source path directive `<path(1)>` is mapped to the reference topic path `b/c/d`. This time the `start` parameter is 1, which corresponds to the topic `b`. Because there is no `number` parameter, the match includes all the topics to the end of the path.

Source value directives

Source value directives enable you to derive the path of the reference topics from the values within a JSON source topic.

Source value directives are only applied to JSON source topics; if the path mapping contains a source value directive, non-JSON topics matching the source topic selector are ignored.

Source value directives use the keyword `scalar` and are parameterized by a single JSON pointer that extracts a scalar value from the source value.

A scalar value is a string, a number, `true`, `false`, or `null`; that is, anything other than an array or a object.

If the JSON pointer does not refer to a scalar value in the source value, no reference topic will be created. This includes the cases where the JSON pointer refers to an array or an object), or when no part of the source value is selected.

Deriving the reference topic paths from part of the source topic value effectively creates a secondary index on the value. For source value directives to work efficiently, the selected scalar values should be relatively stable.

By default, if an update to the source topic changes the selected scalar value, the corresponding reference topic is removed and a new reference topic is created. You can change this behavior using the preserve topics clause (see below).

For example, given a source value of:

```
{
 "account" : "1234",
```

```
 "balance" : { "amount" : 12.57, "currency" : "USD" }
}
```

and a source value directive:

currency/<scalar(/balance/currency)>/account/<scalar(/account)>

the resulting reference topic path will be:

currency/USD/account/1234

By default, if the extracted value is a string, it is copied literally to the reference topic path. A value that contains path separators (`/`) creates a reference topic path with more levels than the path mapping template. You can override this using the separator clause (see below).

An extracted value of `null` will be copied to the reference topic path as the string "null".

Expand value directives

Expand value directives enable you to create multiple reference topics from a single JSON source topic.

Expand value directives are only applied to JSON source topics; if the path mapping contains an expand value directive, non-JSON topics matching the source topic selector are ignored.

Expand value directives use the keyword `expand` and take one or two JSON pointers as parameters:

`<expand(sourcePointer, pathPointer)>`

The `sourcePointer` parameter is a JSON pointer used to specify an array or object to expand.

Every direct child of the indicated source value is expanded into a new reference topic. If no pointer is specified, the value is expanded from the root.

The optional `pathPointer` parameter can be used to name the topic path. A JSON pointer is used to specify a scalar value (that is, not an array or object) within the expanded value. The value is used to derive the reference topic path.

For example, suppose this is the content of a JSON topic called `allCars`:

```
{
 "cars": [
  { "reg":"HY58XPA", "type":"Ford", "model":"Sierra" },
  { "reg":"PY59GCA", "type":"Fiat", "model":"Panda"},
  { "reg":"VA63ABC", "type":"Ford", "model":"Ka"}
  ]
}
```

The following topic view specification:

```
map allCars to cars/<expand(/cars, /reg)>
```

results in these reference topic paths:

```
cars/HY58XPA
cars/PY59GCA
cars/VA63ABC
```

The value of `cars/HY58XPA` is:

```
{
 "reg":"HY58XPA",
 "type":"Ford",
 "model":"Sierra"
}
```

If the second pointer is not specified or no scalar value is found for the pointer, the path fragment is taken from the key (if the child value is an object) or the index (if the child value is an array).

In the example above, if you used the topic view

```
map allCars to cars/<expand(/cars)>
```

, the topic path is taken from the index of the current array element resulting in topics:

```
cars/0
cars/1
cars/2
```

Expand directives can be nested (that is there can be more than one expand directive in a path mapping). In this case, a second expand directive will use the value from the previous expand as its source (root) value and not the value of the source topic. This also applies to scalar directives that follow an expand directive.

Suppose the previous array example is extended so that each car can have multiple drivers:

```
{
 "cars": [
  { "reg":     "HY58XPA",
  "drivers": [{"name" : "Bill"}, {"name" : "Fred"}]
  },
  { "reg":     "PY59GCA",
  "drivers": [{"name" : "Jane"}, {"name" : "Fred"}]
  },
  { "reg":     "VA63ABC",
  "drivers": [{"name" : "Tom"}, {"name" : "John"}]
  }
 ]
}
```

This topic view uses nested expand directives to expand both levels of the array hierarchy:

```
map allCars to cars/<expand(/cars, /reg)>/drivers/<expand(/
drivers, /name)>
```

resulting in these reference topics:

```
cars/HY58XPA/drivers/Bill
cars/HY58XPA/drivers/Fred
cars/PY59GCA/drivers/Jane
cars/PY59GCA/drivers/Fred
```

```
cars/VA63ABC/drivers/Tom
cars/VA63ABC/drivers/John
```

The second expand directive takes the `/drivers` values from the previous expand, so the value of each topic is `{"name" : "Bill"}`, `{"name" : "Fred"}`, `{"name" : "Jane"}` and so on.

If expansion causes more than one mapping to the same topic path, only the first encountered will be created and updated.

Insert clause

The optional insert clause can be used to insert data from topics other than the selected source topics, enabling you to merge data from mutiple topics together.

You can insert data from JSON or scalar (integer/string) topics. The reference topics receiving the data must be JSON topics.

You can either insert the whole value of a topic, or part of a JSON value.

You can specify which part of a JSON topic to insert by using a JSON pointer. You can also use a JSON pointer to specify where to insert within the reference topic.

Insert clauses are specified directly after the path mapping, but before any other clauses to indicate that values from other topics (insertion topics) are to be added to the derived reference topic value. An exception to the rule that they are specified before other clauses is that insert clauses may be interspersed with patch clauses.

The insert clause has the form:

```
insert path_specification key source_key at insertion_key default
constant
```

for example:

```
map ?Some_Source_Topics/ to Mapped_Topics/<path(1)> insert
Some_Other_Topic at /Some_JSON_Pointer
```

path_specification

This specifies the path of the non-source topic from which data is to be obtained and merged with the current input data value (in the simplest case, this is the value from the selected source topic; this can depend on the other clauses).

The path specification uses similar syntax to the path mapping clause.

The specification can contain:

- Constants
- <path()> directives operating on the path of the selected source topic
- <scalar()> directives operating on the current input data value

For example:

Topic/<path(0, 2)>/<scalar(/foo)>

specifies insertion from a topic whose path is `Topic/` followed by the first three elements of the source topic path, followed by the value at the key /foo in the current input data value.

Note that this enables you to select additional nonsource topics to merge based on the structure and content of the source topics.

source_key

This optionally specifies the key (a JSON pointer) of an item within the topic indicated by `path_specification`. If not specified, the whole of the data value of the selected topic will be inserted.

insertion_key

This specifies a JSON pointer indicating the location of the insertion in the current input data value.

Typically this is an object key indicating the key of the value in the data.

If the data already has an item with the same key it is overwritten. Otherwise, a new item is added to the parent indicated by the specified key. If the parent does not exist, the insertion does not occur and a warning is logged.

This can also indicate an entry in an array. If an index key is provided, the existing entry at the specified index is replaced. Use the special `-` character to append to an array. For example, to append to the end of an array at MyArray use `/MyArray/-` as the insertion key.

If the resolved key indicates a scalar item, no insertion will take place.

default constant

Usually if insertion fails, nothing happens. The `default` keyword can be used to override this behavior by providing a default value to insert.

Insertion can fail if:

- the topic specified to insert from (using `path_specification`) cannot be found
- the topic is not a JSON/scalar topic
- the specified key does not exist

If `default` is used, if insertion fails, the specified `constant` value is inserted as a scalar value at the insertion point.

The current input data value is typically the value from the selected source topic.

However, there are some situations where this is not the case:

- if the path mapping includes one or more expand directives, each of the expanded data values is treated as the current input data value in turn. If a path mapping is used to expand an array of five elements, then the insert clause is executed 5 times, once for each element.
- if the insert was preceded by an as<value(key)> directive, the current input data is the data indicated by the key.
- if the insert was preceded by another insert clause, then the current input data is the output from that clause; this enables input clauses to be chained.

Example insert clauses:

```
map Topic1 to Topic2 insert AnotherTopic at /key default "unknown"
```

Map Topic1 to Topic2, and the data within AnotherTopic is inserted into it at the key named other.

```
map ?Topics/ to Mapped/<path(1)> insert AnotherTopic at /other
```

Like the previous example, but all of the topics under the path Topics will be selected and mapped to topics with the same name under the path Mapped. Every selected topic will have the value of AnotherTopic inserted into it (assuming they are JSON objects) unless AnotherTopic does not exist, in which case no insertions would take place.

```
map ?Topics/ to Mapped/<path(1)> insert Others/<path(1)> at /other
```

This uses the more powerful path mapping capabilities of topic inserts. In this case, each selected topic has an insertion from a topic with the same topic under the path Others. For example, Topics/A/B would generate a reference topic at path Mapped/A/B which has the value of Others/A/B inserted at the key other.

```
map ?Topics/ to Mapped/<path(1)> insert Others/<scalar(/foo)> at /other
```

Similar to the previous example, but in this case the path of the insertion topic is derived from a value within the selected source topic. So if topic Topics/A/B has a value of "bar" at key "foo", the topic selected to insert from would be Others/bar.

```
map ?Topics/ to Mapped/<path(1)> insert Others/<path(1)> key /foo at /
other
```

All previous examples have shown the insertion of the whole value of another topic. Here the `key` keyword is used to select a specific item foo within the insertion topic value. If the insertion topic does not have the value with key foo, then a reference topic will be created but no insertion will occur.

When expand directives are used, the insert will occur for every output from the expansion. For example:

```
map Topic1 to Expanded/<expand()> insert AnotherTopic at /other
```

If we assume that the content of Topic1 is an array of objects then each array element would be expanded to produce a new topic at path Expanded/0, Expanded/1 and so on, and each resulting reference topic will have the value from AnotherTopic inserted at the key /other.

Insert clauses can be chained together:

```
map Topic1 to Topic2 insert AnotherTopic at /other insert
YetAnotherTopic at /yetAnother
```

In the above example values from two different topics are inserted into the data to produce the reference topic.

And finally, the insert clause can be used along with `as <value()>` clauses, for example:

```
map Topic1 to Topic2 insert AnotherTopic at /foo/bar as <value(/foo)>
```

In this example the data from AnotherTopic is inserted at the key foo/bar, then the full value of foo is projected.

Patch clause

Patch clauses may be specified directly after the path mapping, but before any other clauses (except insert clauses, which can be interspersed), to indicate that a JSON patch is to be applied to the value.

Patch clauses only apply to JSON source topics; if the topic view specification contains a patch clause, non-JSON topics matching the source topic selector are ignored.

The patch is applied to the current value within a view processing chain, so if a patch occurs after an expand then the patch will be applied to each expanded value.

The format of a patch clause is:

```
patch 'patch string'
```

The patch string should be formatted according to the JSON Patch standard (see RFC 6902: JavaScript Object Notation (JSON) Patch).

Patches are a sequence of JSON Patch operations contained in an array. They are applied as an atomic update to the previous value if the resulting update is successfully calculated. The following patch will check the value at a specific key and update if the expected value is correct: `[{"op":"test", "path":"/price", "value" : 22}, {"op":"add", "path":"/price", "value": 23}]`

The available operations are:

- **Add:** `{"op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ]}`
- **Remove:** `{"op": "remove", "path": "/a/b/c"}`
- **Replace:** `{"op": "replace", "path": "/a/b/c", "value": 43}`
- **Move:** `{"op": "move", "from": "/a/b/c", "path": "/a/b/d"}`
- **Copy:** `{"op": "copy", "from": "/a/b/c", "path": "/a/b/e"}`
- **Test:** `{"op": "test", "path": "/a/b/c", "value": "foo"}`

The test operation checks that the CBOR representation of the value of a topic is identical to the value provided in the patch after converting it to CBOR. If the value is represented differently as CBOR, commonly due to different key ordering, then the patch will return the index of the failed operation.

For example, the values `{"foo": "bar", "count": 43}` and `{"count": 43, "foo": "bar"}` are unequal despite semantic equality due to the differences in a byte for byte comparison.

The following patch clause would add the 'price' field and remove the 'name' field from an input JSON object.

```
 patch '[{"op":"add", "path":"/price", "value" : 22},
{"op":"remove", "path":"/name"}]'
```

Patches can only be applied to JSON arrays or objects and if they fail to apply, no resulting reference topic will be created by the view. If an update patch fails, any previously created reference topic would be removed.

Any number of patch clauses may be chained together and interspersed with insert clauses. For example:

```
patch '[{"op":"add", "path":"/price", "value" : 22}]'
insert AnyTopic at /T/- default "unknown"
patch '[{"op":"remove", "path":"/name"}]'
```

Topic property mapping clause

The topic properties of a reference topic are derived from the source topic. Some topic properties can be changed using the optional topic property mapping clause.

A topic property mapping clause begins with the keywords `with properties` and consists of a comma-separated list of topic property keys and values, each separated by a colon. For example, the following topic view specification maps all topics below the path `a` to reference topics below the path `b`, and disables both conflation and compression for the reference topics.

```
map ?a// to b/<path(1)> with properties CONFLATION:off,
 COMPRESSION:false
```

The following table describes the behavior for each topic property.

Table 11: Reference topic property mapping

Source topic property	Reference topic spec default	Set by topic property mapping?	Notes
COMPRESSION	Copied from source topic specification	Yes	
CONFLATION	Copied from source topic specification	Yes	
DONT_RETAIN_VALUE	Copied from source topic specification	Yes	
OWNER	Copied from source topic specification	No	
PERSISTENT	Not set	No	Reference topics are not persisted. Topic views are persisted, so a reference topic will be recreated on server restart if its source is persistent.
PUBLISH_VALUES_ONLY	Copied from source topic specification	Yes	
REMOVAL	Not set	No	Reference topics cannot be removed directly.
SCHEMA	Copied from source topic specification	No	A RECORD_V2 reference topic has the same schema as its source topic.
SLAVE_MASTER_TOPIC	Not set	No	If a reference topic has a slave topic as its source topic, it indirectly references the slave's master topic.
TIDY_ON_UNSUBSCRIBE	Copied from source topic specification	Yes	
TIME_SERIES_EVENT_VALUE_TYPE	Copied from source topic specification	No	A TIME_SERIES reference topic has the same value type as its source topic.
TIME_SERIES_RETAINED_RANGE	Copied from source topic specification	Yes, with restrictions	A topic property mapping cannot increase the time series retained range by overriding the TIME_SERIES_RETAINED_RANGE property. The retained range of a reference time series topic will be constrained to be no greater than that of its source topic.
TIME_SERIES_SUBSCRIPTION_RANGE	Copied from source topic specification	Yes	

Source topic property	Reference topic spec default	Set by topic property mapping?	Notes
VALIDATE_VALUES	Not set	No	A reference topic reflects updates to its source topic. It cannot reject updates.

Topic value mapping

The value of a reference topic is derived from the source topic according to the topic view value mapping. By default, a reference topic's value is the same as its source topic.

For JSON source topics, the optional topic value mapping clause can be used to extract part of the source value. A topic value mapping begins with the keyword `as` and is followed by a **value directive**.

A value directive is delimited by angle brackets (<, >), and consists of the value keywords and a single JSON pointer parameter. The JSON pointer selects the part of the source value to copy. For example, given a source value of:

```
{
 "account" : "1234",
 "balance" : { "amount" : 12.57, "currency" : "USD" }
}
```

and the value mapping clause as `<value(/balance)>`, the reference topic value will be:

```
{
 "amount" : 12.57,
 "currency" : "USD"
}
```

Topic value mappings are often used with path value mappings. For example:

```
map ?accounts// to balances/<scalar(/account)> as <value(/balance)>
```

Throttle clause

The optional throttle clause can be used to constrain the rate at which a reference topic is updated when its source topic is updated. The primary application of a throttle clause is to restrict the number of updates sent to reference topic subscribers, reducing network utilization or the processing each subscriber must do. Throttling also restricts the rate at which client sessions can observe changes to reference topic values using the fetch API.

The throttle clause has the form:

```
throttle to X updates every period
```

where `X` is a positive integer, and `period` is a positive integer followed by a time unit: `seconds`, `minutes`, or `hours`.

For example, the following topic view specification maps all topics below the path `a` to reference topics below the path `b`, but updates the value of each reference topic at most twice every five seconds:

```
map ?a// to b/<path(1)> throttle to 2 updates every 5 seconds
```

To improve readability, the throttling clause allows `1 update` as an alternative to `1 updates`, and `every second` as an alternative to `every 1 seconds` (as well as for other time units).

For example, the following topic view specification maps all topics below the path a to reference topics below the path b, but updates the value of each reference topic at most once every hour:

```
map ?a// to b/<path(1)> throttle to 1 update every hour
```

The throttle clause is only applied when a source topic is updated more frequently than the configured rate. If a source topic is updated less frequently, updates are passed on unconstrained. If the rate is exceeded, a reference topic will not be updated again until the configured period has expired. At this time, the reference topic will be updated based on the source topic updates that happened in the interim, and a single value will be published. Thus, a throttle clause provides topic-scoped conflation.

The throttle clause is ignored for time series topics because time series updates do not support efficient conflation. Updates to source time series topics are passed on immediately to the corresponding reference topics, regardless of any throttle clause.

Delayed topic views

You can use the optional delay clause to create a delayed topic view. This means that changes to the source topics are reflected in the reference topics after a fixed delay of your choice.

The delay period can be from one second to many days.

A delayed topic view can be useful if you provide real-time data as part of a paid service, and want to provide a delayed feed as a preview or for a lower service tier.

The maximum delay possible is constrained by the disk space available on the Diffusion server to store the delayed updates (in a `delay_store` subdirectory of the persistence directory). The amount of space required will vary depending on the details of your application, such as how many source topics are included in the view and how frequently they update.

The delay does not just apply to topic values: topic additions and removals are also delayed, so the reference topics will reflect the structure of the source topics as they were at the earlier point in time.

The delay clause has the form `delay by duration` where `duration` is a positive integer followed by a time unit which is one of `seconds`, `minutes`, or `hours`.

For example, the following topic view specification maps all topics below the path a to reference topics below the path b, with a five minute update delay.

```
map ?a// to b/<path(1)> delay by 5 minutes
```

Views with delay clauses initially create reference topics in an unpublished state. The topics are published once the delay time has passed.

Note that if you restart the Diffusion server, the delay time will need to pass before any reference topics are published. Delayed events are not persisted.

An unpublished reference topic prevents a lower priority topic view from creating a reference topic with the same path.

Client sessions with the rights to read the source topic can browse unpublished topics using a fetch request.

Preserve topics clause

The preserve topics clause enables you to prevent the default behavior of removing a reference topic with a path derived from a source topic value when that value changes.

For example, given a source value of:

```
{
     "species" : "diplodocus",
```

```
    "exhibit" : { "id" : 137, "category" : "fossil" }
    }
```

and a source value directive:

```
specimen/<scalar(/exhibit/category)>/species/<scalar(/species)>
```

the resulting reference topic path is:

```
specimen/fossil/species/diplodocus
```

What happens when the source topic value changes? By default, the reference topic at the old path is removed, and a new one is created.

In the example above, suppose the value for "species" changes from "diplodocus" to "brontosaurus".

By default, the topic at `specimen/fossil/species/diplodocus` is removed and a new topic is created at `specimen/fossil/species/brontosaurus`.

In some cases, you may not want your application's reference topics to be removed because the source value changes.

The preserve topics clause changes the default behavior so that all reference topics created by a view are retained and updated, until either the source topic or the creating view are removed.

To apply the preserve topics clause, add `preserve topics` to the specification, for example:

```
specimen/<scalar(/exhibit/category)>/species/<scalar(/species)>
preserve topics
```

> **Note:** Currently, preserved topics are not restored if the Diffusion server is restarted.

> **Note:** Currently, preserved topics are not replicated across a cluster. If a new server enters a cluster, it will only have reference topics generated from the point in time where it joined the cluster and will not reflect reference topics previously created within other cluster peers.

Separator clause

Topic views can derive path mappings for reference topics from the value of a source topic. By default, `/` characters in the text are interpreted as path separators and will introduce extra levels in reference topic paths.

For example, consider the topic view:

map ?a/path/ to b/<scalar(/markets/name)>

Suppose the value at /markets/exchange in the source topic is "USD/Sterling". The `/` character is treated as a path separator, so the reference topic is created at `b/USD/Sterling`.

The separate clause enables you to prevent this by specifying a replacement string to be used instead of a path separator in the names of derived reference topics.

In the above example, by adding a separator clause to the topic view:

map ?a/path/ to b/<scalar(/markets/name)> separator '%'

the reference topic is now created at `b/Gold%Silver`.

Note that the separator is a string, so it can be longer than a single character. It can contain `/` characters which will be interpreted as path separators. It must not contain empty path segments, that is `//`.

Type clause

The optional type clause can be used to specify the topic type of the target reference topic. If the current source topic type can be converted to the target type, a reference topic of the specified type will be created.

The 'type' clause can be used anywhere after path mapping or insert clauses.

For example:

```
map ?a/ to b/<path(1)> type STRING
```

The specified type must be one of: STRING, INT64, DOUBLE, JSON, TIME_SERIES or BINARY.

The following table describes the supported conversions from the source topic type (the left column) to the supported target types. If the source topic type is not listed in the left column, no conversion is supported; topics of unsupported types are not processed.

In all cases, the value being processed is the current value as derived from other mappings within the topic view (for example, expand), which is not necessarily the value of the source topic.

Table 12: Reference topic type mapping

Source topic	Reference topic					
	STRING	INT64	DOUBLE	JSON	TIME_SERIES	BINARY
STRING	-	Conversion occurs if string's value can be converted to target number type. If string cannot be converted, no reference topic is created.		Creates JSON topic containing the scalar value.	Creates a time series with event type matching source. Source topic updates are appended to reference topic.	x
INT64	Conversion from number types to STRING results in a simple string representation of the number.	-	INT64 to DOUBLE conversions perform a simple conversion. For example, 123 becomes 123.0.	Creates JSON topic containing the scalar value.	Creates a time series with event type matching source. Source topic updates are appended to reference topic.	x
DOUBLE	Conversion from number types to STRING results in a simple string representation	DOUBLE to INT64 conversions perform rounding to the nearest integer value. For	-	Creates JSON topic containing the scalar value.	Creates a time series with event type matching source. Source topic updates are	x

Source topic	Reference topic					
	of the number.	example 12.51 becomes 13.			appended to reference topic.	
JSON	JSON to primitive type conversions only occur if the JSON value is a scalar which can be read as a string and converted to the target type. Currently, string and integer scalar values can be read as a string, but not doubles. The string representation is converted in the same way as specified for STRING to other primitive types. If the JSON value is a structure or cannot be converted, no reference topic is created.			-	Creates a time series with event type matching source. Source topic updates are appended to reference topic.	x
TIME_SERIES	The conversion of TIME_SERIES to other types follows the rules for the source topic type that matches the time series topic's event type. So if the time series event type is 'double', the conversion rules from DOUBLE to the target type apply. Each value appended to the source time series topic results in an update to the reference topic. If conversion fails at any point, the reference topic is removed and only recreated if a value that can be converted is appended.				-	BINARY to TIME_SERIES and TIME_SERIES to BINARY are supported in the same way as for other time series conversions.
BINARY	x	x	x	x	BINARY to TIME_SERIES and TIME_SERIES to BINARY are supported in the same way as for other time series conversions.	-

Note: An 'x' indicates that a conversion is not supported.

Remote topic views

A remote topic view maps its source topics from a remote server.

This is indicated with a `from` clause:

```
map ?a/ from Server1 to b/<path(1)>
```

The above example maps the topics under topic node `a` on a remote server to topic node `b` on the server where the topic view was created.

`Server1` is the name of a remote server definition, created using any client API.

If a topic view is created with a remote server definition that does not exist, the topic view will remain inactive until the remote server definition is created and a connection is established to the remote server.

Any number of topic views can share a remote server definition and map topics from the same server.

If a remote server definition is removed when one or more topic views refer to it, reference topics created by those views will be removed and the views will become inactive. If the remote server is recreated, the views will reactivate. This ensures that network connections are only in place for remote servers when there are topic views that require them.

When a connection to a remote server connection is lost, the reference topics of all topic views that refer to it will be removed.

Normal topic view precedence rules are applied to remote views. If two topic views try to create reference topics on the same path, the view that was created first takes precedence, whether it is local or remote.

If a remote view has no working server connection, it will not prevent a younger local view creating a reference topic on the same path; but if the remote view then establishes a connection, precedence will be applied, and the reference topic from the local view is replaced by one mapped from the remote view.

In a Diffusion cluster, both topic views and remote server definitions are automatically distributed across the cluster. Each member of the cluster will automatically connect to the same remote server and produce the same reference topics from the remote topic views.

All of the other topic view capabilities can be applied to a remote topic view: for example, a remote topic view can have an expand value directive or a throttle clause.

Quoting and white space

Topic selectors and path mapping templates can be quoted or unquoted. They are quoted using the single quote mark. To include whitespace, single quotes or literal opening angle brackets they must be quoted. In quoted selectors and templates single quotes, literal opening angle brackets and backslashes must be escaped with a single backslash. In templates the opening angle bracket should be unescaped when beginning a directive. Characters in unquoted selectors and templates can't be escaped.

Any whitespace can be used to separate keywords, statements and clauses.

Dealing with topic path conflicts

If you create a topic view which tries to make a reference topic with the same path as an existing topic, the result is a topic path conflict.

Reference topics have a lower priority than normal topics created through the API, including replicas of normal topics created by topic replication or fan-out. A reference topic will only be created if no topic or reference topic is already bound to its derived topic path.

Topic views have a precedence based on order of creation. If two topic views define mappings to the same topic path, the earliest-created topic view will create a reference topic. If a topic view is updated, it retains its original precedence.

Topic view persistence and replication

Reference topics are neither replicated nor persisted. They are created and removed based on their source topics. However, topic views are replicated and persisted. A server that restarts will restore topic views during recovery. Each topic view will then create reference topics based on the source topics that have been recovered.

The server records all changes to topic views in a persistent store.

Topic views are restored if the server is started. If a server belongs to a cluster, topic views will be replicated to each server in the cluster. Topic views are evaluated locally within a server. Replicated topic views that select non-replicated source topics can create different reference topics on each server in the cluster.

Access control

The following access control restrictions are applied:

- To list the topic views, a session needs the READ_TOPIC_VIEWS global permission.
- To create, replace, or remove a topic view, a session needs the MODIFY_TOPIC_VIEWS global permission and SELECT_TOPIC permission for the path prefix of the source topic selector.
- Each topic view records the principal and security roles of the session that created it as the topic view security context. When a topic view is evaluated, this security context is used to constrain the creation of reference topics. A reference topic will only be created if the security context has READ_TOPIC permission for the source topic path, and MODIFY_TOPIC permission for the reference topic path. The topic view security context is copied from the creating session at the time the topic view is created or replaced, and is persisted with the topic view. The topic view security context is not updated if the roles associated with the session are changed.

Related concepts

Topic tree on page 42
Diffusion primarily distributes data using a pub-sub model, where content is published to topics. These topics are arranged as a tree.

Topic selectors on page 44
A topic selector defines a set of topics paths that identify topics. You can create a topic selector from a topic selector expression.

Session trees

A session tree is a virtual view of the topic tree presented to a session by fetch and subscription operations. A custom session tree can be configured and maintained based on a session's properties. This can be useful for security, optimisation, personalisation and localisation.

Session trees overview

Session trees enable your application to present tailored data to each client session. By using session trees, you can present each session with a subset of the full topic tree based on its session properties. This enables tailored data to be served to a session even if it is not authenticated. For example, you could localize your application by showing different data to a session based on its $Country session property.

Session trees are configured using a set of declarative rules maintained by the server. You can create or change the rules using one of the client SDKs.

Session trees complement the data transformation capabilities of topic views. Topic views enable you to create new branches of the topic tree from existing branches; session trees enable you to tailor which branches a session can access.

Branch mappings and session paths

A session tree is produced by applying branch mappings to the topic tree. Branch mappings are organised into branch mapping tables. Each branch mapping table is assigned to a unique path: the session tree branch.

A session tree is composed of session paths. Each session path is mapped via the branch mapping tables to a unique topic path.

A branch mapping table is an ordered list of (session filter, topic tree branch) pairs. For example, the branch mapping table for the session tree branch `market/prices` might be:

Table 13: Branch mapping example

Session filter	Topic tree branch
USER_TIER is '1' or $Country is 'DE'	backend/discounted_prices
USER_TIER is '2'	backend/standard_prices
$Principal is ''	backend/delayed_prices

With this configuration, if an unauthenticated session (one that matches the `$Principal is ''` session filter) subscribes to the session path `market/prices/X`, and there is a topic bound to the topic path `backend/delayed_prices/X`, the subscription will complete. The session will receive a subscription notification under the session path `market/prices/X`, together with the topic properties and the value of the topic. The session is unaware that the data originates from a topic bound to a different topic path. If no topic is bound to `backend/delayed_prices/X`, the subscription will not resolve and the session will receive no data, even if there is a topic bound to `market/prices/X`.

Session trees complement the data transformation capabilities of topic views. In our example, the time delayed time feed at backend/delayed_prices could be maintained by a topic view using the delay by clause.

Branch mappings are persisted by the server and shared across a cluster, in a similar manner to topic views, security stores, and metric collectors. Branch mappings are editable using this feature, and via the management console.

For a given session and session path, at most one branch mapping applies. The applicable branch mapping is chosen as follows:

- Each branch mapping table with session tree branch that is a prefix of the session path is considered. For a given table, the first branch mapping with a session filter that matches the session's properties is the one that applies. A branch mapping table may have no applicable branch mappings for a session.
- If there are several such branch mapping tables with a branch mapping that for the session, the one with the longest prefix of the session path applies.
- If no branch mapping table has a branch mapping for the session, the session path is translated to the identical topic path.

Access control

To subscribe to or fetch from a session path, a session must be granted the appropriate path permission to the session path for the operation (SELECT_TOPIC, or READ_TOPIC). The session doesn't require any permissions to the topic path of the topic providing the data.

To create or replace branch mappings, a session needs the MODIFY_TOPIC path permission for the session tree branch of the branch mapping table, EXPOSE_BRANCH path permission for the topic tree branch of each branch mapping, and (if an existing table with the same session tree branch is being replaced) EXPOSE_BRANCH permission for each branch mapping of existing table.

To retrieve a branch mapping table, a session needs the READ_TOPIC path permission for its session tree branch.

Topics

Consider the types of topic you want to use and how.

A topic is a channel through which data can be distributed to clients. Topics provide a logical link between publishing clients and subscribing clients. For more information, see Pub-sub on page 93.

Diffusion topics are typed. Data sent through topics must match the data type of the topic.

Topics that store values

You can publish data to these topics and the data is streamed to subscribing clients. These topics provide a range of possible data types:

- JSON
- Binary
- String
- Int64
- Double
- RecordV2

Advanced topics

Diffusion includes advanced topic types that provide additional capabilities such as storing a series of events or routing subscriptions to other topics.

- Time series
- Routing
- Slave (DEPRECATED)

Properties

You can also assign properties to topics when you create them. The properties a topic can have can change depending on the topic type. For more information, see Properties of topics on page 69.

Properties of topics

When you create a topic, you can specify properties that the topic has. The available properties depend on the topic type.

The following table shows which properties are available. Some property names have been abbreviated; see below the table for the full names.

Table 14: Properties available for topics of each type

Typ	PUBLIS VALUES ONLY	DONT RETAIN VALUES	SLAVE MASTER TOPIC (DEPRE	TIDY ON UNSU	TIME SERIES EVENT VALUE TYPE	TIME SERIES RETAIN RANGE	TIME SERIE SUBS(RANG	VALIDA' VALUES	SCH	COI	COI	OW	REI	PEF	PRI
JSON	✓	✓		✓				✓		✓	✓	✓	✓	✓	✓
Binary	✓	✓		✓				✓		✓	✓	✓	✓	✓	✓

Typ	PUBLIS VALUES ONLY	DONT RETAIN VALUES	SLAVE MASTEF TOPIC (DEPRE	TIDY ON UNSU	TIME SERIES EVENT VALUE TYPE	TIME SERIES RETAIN RANGE	TIME SERIE SUBS(RANG	VALIDA VALUES	SCH	COI	COI	OW	REI	PEF	PRI
String	✓	✓		✓				✓		✓	✓	✓	✓	✓	✓
Int64		✓		✓				✓		✓	✓	✓	✓	✓	✓
Double		✓		✓				✓		✓	✓	✓	✓	✓	✓
Time series	✓	✓		✓	✓ REQUIRED	✓	✓	✓		✓	✓	✓	✓	✓	✓
Routing				✓											
Slave (DEPRECATED)			✓ REQUIRED	✓											
RecordV2		✓		✓				✓	✓	✓	✓	✓	✓	✓	✓

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

SLAVE_MASTER_TOPIC (DEPRECATED)

This property is now deprecated and will be removed in a future release.

The path to the topic that acts as the master topic to a slave topic. A topic is not required to exist at this path at the time the slave topic is created.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

TIME_SERIES_EVENT_VALUE_TYPE

Set this to the type name of a Diffusion data type. All events in the time series are of this data type. The type name can be one of the following values:

- json
- binary
- string
- int64
- double
- record_v2

TIME_SERIES_RETAINED_RANGE

Set this to a range expression that specifies the range of events retained by a time series topic. When a new event is added to the time series, older events that fall outside of the range are discarded. If the property is not specified, a time series topic retains the ten most recent events.

For more information about range expressions, see Range expressions on page 73.

TIME_SERIES_SUBSCRIPTION_RANGE

Set this to a range expression that specifies the range of events sent to all new subscribers.

If a range expression is specified for this property, the specified subscription range is sent to the client session. This is true whether delta streams are enabled for the topic or not. However, to receive all the events in the specified range, the subscribing client session must register a stream before it subscribes to the topic. If a stream is not registered before subscription, the session receives only the latest value.

If the property is not specified, new subscribers will be sent the latest event if delta streams are enabled for the topic and no events if delta streams are disabled for the topic.

For more information about range expressions, see Range expressions on page 73.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

SCHEMA

Optionally, define valid records and fields of a recordV2 topic.

A recordV2 topic contains records, which can be divided into fields. The schema names the records and fields and provides a mechanism for direct access to the fields. The schema is also used to validate the data to ensure it complies with the schema definition. The schema property is supplied as a JSON string that can be generated from a Schema object.

If no schema is provided, the topic data can be free format.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

COMPRESSION

Used to set a level of automatic compression for the topic.

The supported values are:

- off
- low
- medium
- high

The default is "low".

Higher compression levels decrease the amount of data, at the expense of greater CPU cost.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

PRIORITY

Used to set a topic delivery priority.

This property is used to prioritise updates sent to a subscribed client session in the event of a backlog.

Priority is applied to topic updates and topic subscription/unsubscription notifications.

The supported values are:

- low
- default
- high

If no priority is set, "default" is used.

See Topic delivery priority on page 121 for more details.

Range expressions

A range expression can contain the following constraints:

A `limit` constraint

A limit constraint specifies the maximum number of events from the end of the time series.

For example, the following expression requests the five most recent events:

```
limit 5
```

A `last` constraint

A last constraint specifies the maximum duration of events from the end of the time series. The duration is expressed as an integer followed by one of the following time units:

- `MS` – milliseconds
- `S` – seconds
- `H` – hours

For example, the following expression requests all recent events that are no more than 30 seconds older than the latest event:

```
last 30s
```

Both a `limit` and a `last` constraint

For example, the following expression requests the ten most recent events that are no more than one minute older than the latest event:

```
last 60s limit 10
```

If a range expression contains multiple constraints, the constraint that selects the smallest range is used.

Range expressions are not case sensitive.

JSON topics

A topic that provides data in JSON format. The data is transmitted in a binary form for increased efficiency and can be transmitted as a structural delta to reduce the amount of data sent. JSON topics are stateful: each topic stores a JSON value the Diffusion server.

Why use a JSON topic?

JSON is a human-readable, industry-standard format for your data. JSON is natively supported by JavaScript and there are third-party libraries available for other platforms. For more information about JSON, see http://www.json.org/.

A JSON topic enables multiple fields to be maintained in the same topic as part of a composite data type. All updates made at the same time to parts of a JSON topic are sent out to the client together. This enables a set of parts to be treated as a transactional group.

Deltas of change are calculated at the Diffusion server such that only those parts that have changed since the last update are sent out to the subscribed clients. These structural deltas ensure that the minimum amount of data is sent to clients.

The current value of the topic is cached on the client. When deltas are sent, the client can automatically apply these deltas to the value to calculate the new value.

As of Diffusion 6.4, JSON topics support partial updates using the JSON Patch standard. This enables a client to update part of the data within a topic, without needing to know the full value. In addition, a partial update can provide performance benefits if the JSON value is large.

If your data structure is too complex to be represented by a topic tree or might make the topic tree structure difficult to manage, it might be more appropriate to represent part of the data structure inside a JSON topic as JSON objects.

The value of the topic is transmitted as CBOR. For more information about CBOR, see http://cbor.io/.

The value of the topic is accessible both as JSON and CBOR.

Properties of a JSON topic

When you create a JSON topic you can specify the following properties in the topic specification:

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a JSON topic

It is possible to store any data in a JSON topic, even if it is not valid JSON. However, this can cause problems when clients receive unexpected values.

You can validate JSON in the client, or on the server by setting VALIDATE_VALUES to true.

All languages parse JSON as text. Only JavaScript has native support for parsing JSON. Other languages must use third-party libraries. The JavaScript clients provide facilities for converting JSON text to and from a Diffusion JSON value.

Only JSON topics support partial updates.

The topic view expansion feature can only be used with JSON topics.

Diffusion JSON values are transmitted as a CBOR representation. Applications can also access the CBOR binary data directly, and it is often more efficient to do so than to first convert the data to JSON text. For example, Java applications can use the third party Jackson library to map the CBOR data directly to Java objects.

Binary topics

A topic that streams binary data as bytes and uses efficient binary deltas to stream only the data that changes between updates. Binary topics are stateful: each topic stores a binary value on the Diffusion server.

Why use a binary topic?

You can use a binary topic to transmit any arbitrary binary data without the overhead of encoding it to a string or the risk of the binary data being incorrectly escaped.

Binary topics can use binary deltas to send only the data that has changed when this is more efficient than sending the full value.

You can use a binary topic to transmit very large strings. This enables a client to use the binary delta capability to transmit only the changed parts of a string rather than the whole value. This reduces the amount of data transmitted over the network.

Binary formats, such as Google protocol buffers, can be streamed using a binary topic.

Properties of a binary topic

When you create a binary topic you can specify the following properties in the topic specification:

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

Binary values are always valid so setting this property has no effect.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a binary topic

Data on binary topics contains no implicit information about its structure.

Data on binary topics cannot be viewed in the console.

String topics

A topic that provides data in string format. The data is transmitted in a binary form for increased efficiency. String topics are stateful: each topic stores a JSON value on the Diffusion server.

Why use a string topic?

A string topic enables you to explicitly type the data that you send through the topic as a string.

String topics support null data values.

Deltas of change are calculated at the Diffusion server such that only those parts that have changed since the last update are sent out to the subscribed clients. This ensures that the minimum amount of data is sent to clients.

The current value of the topic is cached on the client. When deltas are sent, the client can automatically apply these deltas to the value to calculate the new value.

The value of the topic is stored and transmitted as a CBOR-encoded string, or CBOR `Null`. CBOR encodes strings using UTF-8. For more information about CBOR, see http://cbor.io/.

Properties of a string topic

When you create a string topic you can specify the following properties in the topic specification:

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a string topic

Int64 topics

A topic that provides data in 64-bit integer format. The data is transmitted in a binary form for increased efficiency. Int64 topics are stateful: each topic stores a 64-bit integer on the Diffusion server.

Why use an int64 topic?

An int64 topic enables you to explicitly type the data that you send through the topic as a 64-bit integer.

Int64 topics support null data values.

The value of the topic is transmitted as CBOR. For more information about CBOR, see http://cbor.io/.

The value of the topic is accessible both as a 64-bit integer and CBOR.

Properties of an int64 topic

When you create an int64 topic you can specify the following properties in the topic specification:

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using an int64 topic

Deltas are not available for data published to int64 topics.

Because of a limitation of the JavaScript platform, the validity of the values converted to an `Int64` from a `Number` or converted to a `Number` from an `Int64` can only be guaranteed for values up to 2^{53} - 1. Converting between a `String` and an `Int64` has fully guaranteed precision.

Double topics

A topic that provides data in double precision floating point format (IEEE 754). The data is transmitted in a binary form for increased efficiency. Double topics are stateful: each topic stores a double value on the Diffusion server.

Why use a double topic?

A double topic enables you to explicitly type the data that you send through the topic as a double precision floating point number.

Double topics support null data values.

The value of the topic is transmitted as CBOR. For more information about CBOR, see http://cbor.io/.

The value of the topic is accessible both as a double and CBOR.

Properties of a double topic

When you create a double topic you can specify the following properties in the topic specification:

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a double topic

Deltas are not available for data published to double topics.

Time series topics

A time series topic holds a sequence of events.

Note: Time series topics are supported by the JavaScript, Java, Android, .NET and Apple APIs.

Why use a time series topic?

A time series topic holds a sequence of events. Time series topics are useful for collaborative applications such as chat rooms. Multiple users can concurrently update a time series topic.

Each event in a time series topic has a value. Each time series topic has an associated data type which determines what type of value its events have: binary, double, int64, JSON, string or recordV2.

All the events in a given time series topic must have the same value type.

Each event is assigned metadata by the Diffusion server. This metadata consists of a sequence number, a timestamp, and an author (where the author is the principal used by the client session that creates the event).

A time series is an append-only data structure. A session can update the topic by providing a new event, which the server will append to the end of the time series. A session can also edit an existing event in time series, providing a new value. Editing an event does not remove or modify the original event, but instead appends an edit event to the end of the time series. Edit events have an additional metadata field, the sequence number of the original event, allowing subscribers to associate the new value with the replaced value.

You can subscribe to events that are published to a time series topic and receive updates in the same way as for other topics. New subscribers also receive an initial subset of the most recent events. You can configure a subscription range to control how far back in time the initial set goes.

You can query a time series topic to receive a range of events based on the timestamp or sequence number of the events in a series.

By default, queries return a merged view of a time series that includes edit events in the place of the original events. A session with the QUERY_OBSOLETE_TIME_SERIES_EVENTS permission can submit a modified query which returns an unmerged view that includes both original events and the edit events that replace them.

Properties of a time series topic

When you create a time series topic you must specify the following property in the topic specification:

TIME_SERIES_EVENT_VALUE_TYPE

Set this to the type name of a Diffusion data type. All events in the time series are of this data type. The type name can be one of the following values:

- json
- binary
- string
- int64
- double
- record_v2

You can also specify the following optional properties in the topic specification:

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

TIME_SERIES_RETAINED_RANGE

Set this to a range expression that specifies the range of events retained by a time series topic. When a new event is added to the time series, older events that fall outside of the range are discarded. If the property is not specified, a time series topic retains the ten most recent events.

For more information about range expressions, see Range expressions on page 73.

TIME_SERIES_SUBSCRIPTION_RANGE

Set this to a range expression that specifies the range of events sent to all new subscribers.

If a range expression is specified for this property, the specified subscription range is sent to the client session. This is true whether delta streams are enabled for the topic or not. However, to receive all the events in the specified range, the subscribing client session must register a stream before it subscribes to the topic. If a stream is not registered before subscription, the session receives only the latest value.

If the property is not specified, new subscribers will be sent the latest event if delta streams are enabled for the topic and no events if delta streams are disabled for the topic.

For more information about range expressions, see Range expressions on page 73.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a time series topic

All the events in a time series topic are stored in memory on the Diffusion server. If the Diffusion server is restarted, the events in the time series are lost unless topic persistence is enabled.

A time series topic retains a range of the most recent events. Older events are discarded. By default, only the ten most recent events are retained (this includes edit events). You can configure the "retained range" property to retain more events.

If you are considering using PUBLISH_VALUES_ONLY for a time series topic, use DONT_RETAIN_VALUE, which has the same effect of disabling delta streaming, but additionally saves memory.

A query against a time series topic only returns edit events that refer to original events in the view range of the query. If the original event is no longer stored on the server due to the retained range, related edit events will never be returned. These "orphaned" edit events will stay on the server until more events have been appended and they are pushed out of the retained range.

If subscribing to a time series topic and using a value stream to receive data, ensure that the client adds the value stream before subscribing to the topic to receive all events in the configurable window. If the client session adds the value stream after it subscribes to the time series topic, the client session only receives the latest event on the time series topic.

Related concepts

Using time series topics on page 297
A client can subscribe to a time series topic using a value stream, query to retrieve values within a range, append new values, or apply an edit event to override the value of an earlier event.

Routing topics

A special type of topic, which can map to a different real topic for every client that subscribes to it. In this way, different clients can see different values for what is effectively the same topic from the client point of view.

Note: The routing topic type is deprecated. Use session trees instead.

When a client subscribes to a routing topic, the request is either passed to a client that has registered as a routing subscription handler for the topic or handled by a server-side routing handler. The routing handler assigns a linked topic to represent it to that client.

The routing handler can assign a different linked topic to each client that subscribes to the routing topic.

When updates are received on the linked topic, those updates are propagated through the routing topic to the subscribing clients.

The subscribing client is not aware of the linked topic. It is subscribed to the routing topic and all the updates that the client receives contain only the routing topic path and not the linked topic path.

Why use a routing topic?

Use routing topics when you want your subscribing clients to all have the same subscription behavior, but the data they receive to be decided by a routing handler depending on criteria about that client.

For example, your subscribing clients can subscribe to a routing topic called Price, but the routing handler assigns each client a different linked topic depending on the client's geographic location. This way, clients in different countries can act in the same way, but receive localized information.

Properties of a routing topic

When you create a routing topic you can specify the following properties in the topic specification:

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

Considerations when using a routing topic

Using routing topics requires that you write a routing handler that is either hosted on the server or registered by a client with the required permissions. The following client APIs can register a routing handler: Java, .NET, or Android API.

A subscribing client only needs permission to subscribe to the routing topic. Permission to subscribe to the linked topic is not required.

If the linked topic is removed, subscribing clients are automatically unsubscribed from the routing topic.

If you attempt to fetch from a routing topic that routes to a stateless topic, no data is returned.

You cannot use topic replication to replicate routing topics between Diffusion servers.

When using automatic fan-out to propagate topics from a primary server to one or more secondary servers, the routing subscription handlers for a routing topic must be registered at the primary and all secondary servers. The routing logic provided by the handlers on the primary and secondary server must be identical.

Slave topics (DEPRECATED)

A special type of topic that has no state of its own but is a reference to the state of another topic.

> **Note:** Slave topics are now deprecated and will be removed in a future version. Use Topic views on page 50 instead of slave topics. Topic views were added in Diffusion 6.3 and can do everything slave topics can do, with more powerful mapping options than are available between master and slave topics.

A slave topic acts as an alias to another topic, the master topic. Updates published to the master are fanned out to subscribers of the slave. The slave cannot be updated directly. The master topic can be any topic type except:

- slave
- routing

The link between a slave topic and a master topic is defined when the slave topic is created. This is different to routing topics where the link between topics is defined when a client session subscribes.

If the master topic does not exist when the slave topic is created, the slave topic is created as an unbound slave topic that is not visible to subscribers. When a topic is created at the master topic path, the slave topic becomes bound and can be subscribed to by client sessions.

Properties of a slave topic

When you create a slave topic you must specify the following property in the topic specification:

SLAVE_MASTER_TOPIC (DEPRECATED)

This property is now deprecated and will be removed in a future release.

The path to the topic that acts as the master topic to a slave topic. A topic is not required to exist at this path at the time the slave topic is created.

When you create a slave topic you can specify the following optional properties in the topic specification:

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

Why use a slave topic?

You can use slave topics to provide the same data from multiple topic paths and manage the topics from only one topic.

You can use a slave topic to act as a redirection to a succession of master topics. For example, you can create a slave topic called latest that is linked to a master topic where data is published about a current event. When that event is no longer current, you can remove the slave topic and recreate it now linked to the master topic where data is published about what is now the current event.

The subscribing client sessions can subscribe to the latest slave topic and they continue to be subscribed to the slave topic and receive the latest data, even as the master topic that provides the data changes.

Considerations when using a slave topic

A client only needs permissions on the slave topic. Permission to subscribe to the linked topic is not required.

More than one slave can point to the same master topic.

A slave topic cannot also act as a master topic to another slave topic.

Removing a master topic causes all linked slave topics to become unbound and any clients that are subscribed to the slave topic become unsubscribed. If a new master topic is created at the linked path, the slave topic is bound and the clients are resubscribed.

When using topic replication to replicate slave topics between Diffusion servers, be aware that a replicated slave topic is linked to a master topic located on the same Diffusion server as the replicated slave topic. This is true whether that master topic is created by replication or directly.

RecordV2 topics

A topic that streams data in recordV2 format, where the data is divided into multiple records, each of which can contain multiple fields. RecordV2 topics are stateful: each topic stores a value consisting of one or more records on the Diffusion server.

You can optionally define the format of the data by using a schema that is associated with the recordV2 topic. You can use a schema to validate the data at the server. A schema also provides a convenient way of building a topic value using the fields defined within the schema, or interpreting a topic value in terms of named records and fields. For more information, see RecordV2 schema on page 91.

If you choose not to provide a schema, the data is treated as free format: the meaning of each field is up to your application, and Diffusion does not perform any validation.

.

Why use a recordV2 topic?

A recordV2 topic enables multiple fields to be maintained in the same topic as part of a composite data type. All updates made at the same time to fields on a record topic are sent out to the client together. This enables a set of fields to be treated as a transactional group.

Deltas of change are calculated at the server such that only those fields that have changed since the last update are sent out to the subscribed clients. This ensures that the minimum amount of data is sent to clients.

Properties of a recordV2 topic

When you create a recordV2 topic you can specify the following properties in the topic specification:

PUBLISH_VALUES_ONLY

By default, delta streaming is enabled. If this property is set to `true`, delta streaming is disabled and all values are published in full.

If there is little or no relationship between one value published to a topic and the next, delta streams will not reduce the amount of data transmitted. For such topics, it is better to set PUBLISH_VALUES_ONLY.

DONT_RETAIN_VALUE

If set to true, the latest value of the topic is not retained by the Diffusion server or the client that publishes it. New clients that subscribe do not receive an initial value. No value will be returned for fetch operations that select the topic.

For time series topics, if DONT_RETAIN_VALUE is set to true, time series events are still retained, but the latest value is not stored separately.

The DONT_RETAIN_VALUE property is useful for applications like a feed of news items, or for values that are only valid at the moment of publication. You can combine this with VALIDATE_VALUES.

Using DONT_RETAIN_VALUE reduces the topic memory footprint, but disables delta streaming. Using an update stream will send complete values to the server, instead of deltas. Disabling delta streaming is likely to increase the bandwidth used unless subsequent values are unrelated.

This property replaces the obsolete stateless topic type which was removed in Diffusion 6.2.

TIDY_ON_UNSUBSCRIBE

If set to `true`, when a session unsubscribes from the topic, any updates for the topic that are still queued for the session are removed.

There is a performance overhead to using this option as the client queue must be scanned to find topic updates to remove, however it may prove useful for preventing unwanted data being sent to sessions. This property is disabled by default.

SCHEMA

Optionally, define valid records and fields of a recordV2 topic.

A recordV2 topic contains records, which can be divided into fields. The schema names the records and fields and provides a mechanism for direct access to the fields. The schema is also used to validate the data to ensure it complies with the schema definition. The schema property is supplied as a JSON string that can be generated from a Schema object.

If no schema is provided, the topic data can be free format.

VALIDATE_VALUES

If set to `true`, the topic rejects updates that would create invalid instances of the topic's data type.

If set to anything other than `true`, no validation is performed and all values are streamed to subscribing clients whether they are valid data or not.

Validation has a performance overhead and is disabled by default.

> **Note:** If validation is disabled and the value provided is not valid, the client might produce errors or other unexpected behavior. The exact error varies depending on the client platform. To avoid this, use the client-side validation method provided by the Diffusion API.

CONFLATION

Used to set a conflation policy for the topic. Conflation can merge or discard topic updates to reduce server memory and network data usage.

The supported values are:

- off
- conflate
- unsubscribe
- always

The default is "conflate".

See Using conflation on page 102 for details of the policies.

OWNER

Used to set a security principal as the owner of the topic. The principal receives `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` privileges.

The format of the property value is:

```
$Principal is "name"
```

where **name** is the name of the principal.

REMOVAL

Used to set a topic removal policy.

The property is an expression which defines the conditions which will trigger automatic removal of the topic or of a set of topics specified with an optional selector.

See Removing topics automatically on page 284 for details of the expression format.

PERSISTENT

Used to disable persistence for a topic.

If set to "false", the topic will not be persisted, even if persistence is enabled for the server.

Considerations when using a recordV2 topic

RecordV2 replaced the older record topic type, which was removed as of Diffusion 6.2. You can migrate your applications from record topics to recordV2 topics with minimal changes. Using recordV2 instead of the obsolete record type is required to use topic persistence and topic replication.

The data within a recordV2 topic can either be free format or constrained by a schema. If no schema property is specified, the topic will be treated as free format.

Update a recordV2 topic with a value updater. See Updating topics on page 291 for more information about value updaters.

Receive data from a recordV2 topic with a value stream. See Using streams for subscription on page 227 for more information.

Related concepts

RecordV2 schema on page 91
A schema is an optional way to define how data is formatted when it is published on a recordV2 topic. A schema defines and names the permitted records and fields within the topic, and enables direct access to the fields.

Update recordV2 topics on page 270
The following example demonstrates how to create and update recordV2 topics, including the use of a schema.

Subscribe to recordV2 topics on page 275
The following example demonstrates how to process information from subscribed recordV2 topics, including the use of a schema.

Related tasks

Defining a recordV2 schema on page 266
You can use the API to specify a schema that defines the content of a recordV2 topic.

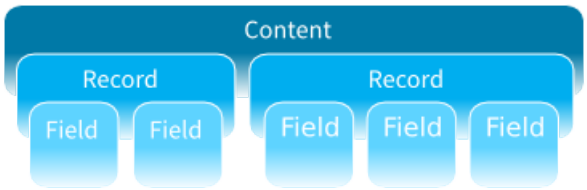
RecordV2 schema

A schema is an optional way to define how data is formatted when it is published on a recordV2 topic. A schema defines and names the permitted records and fields within the topic, and enables direct access to the fields.

The recordV2 topic type contains data organized into records and fields. You can optionally provide a schema which defines the expected structure of the data.

RecordV2 structure

The recordV2 topic type has a value consisting of records, which contain fields. Each recordV2 topic can contain one or more *records*. Each record can contain one or many *fields*.



Using a schema

With a schema, you can define how the records and fields within a recordV2 topic are laid out.

Fields and records within a schema are identified by a name. Every record must have a name that is unique within the content. Every field must have a name that is unique within the enclosing record.

Every field or record defined in the schema can represent one or more possible occurrences of that field or record in the data. The number of possible occurrences of a record or field is described by its multiplicity.

The order in which records and fields are defined within the schema sets the order that they appear within the topic.

Records

A record can contain one or more fields.

Every record has multiplicity.

Fields

A field defines an elementary data item within a record.

Every field has the following properties:

- Multiplicity
- Data type

Multiplicity

The multiplicity of a field or record in a schema defines the number of times it can occur in the topic. Multiplicity is set by providing a minimum value and a maximum value.

Fixed multiplicity means the minimum and maximum are the same. For example, if a field has a minimum of 5 and a maximum of 5, there must be exactly five occurrences of the field within its enclosing record.

Variable multiplicity means the minimum and maximum are different. For example, a schema could specify that there must be between one and five occurrences of a field within its enclosing record. Variable multiplicity is only allowed in the last record in a topic, or the last field in a record.

Use a minimum value of 0 to define an optional field/record. A fixed multiplicity of 0 is not allowed.

A maximum value of -1 is used to represent that there is no limit to how many times the field or record can occur.

Data type

The data type of a field defines the type of values it can contain. The following table describes the data types that are available.

Table 15: Data types for schema fields

Data type	Description
String	A character string.
Integer	An integer represented in the content as a character string. If a field is defined as this type, it can only contain numeric digits with an optional leading sign. Fields of this type cannot be empty.

Data type	Description
Decimal	A decimal number represented in the content as a character string. Decimal fields have the number of places to the right of the decimal point defined by the scale. Such values can be parsed from a character string with any number of digits to the right of the decimal point. Half-up rounding is applied to achieve the target scale. Output of the field is rendered with the specified scale. Fields of this type cannot be empty. For comparison purposes the scale is ignored: a value of 1.50 is the same as 1.5.

Defining a schema

See Defining a recordV2 schema on page 266 for details of how to define a schema and apply it to a recordV2 topic.

Related concepts

RecordV2 topics on page 89
A topic that streams data in recordV2 format, where the data is divided into multiple records, each of which can contain multiple fields. RecordV2 topics are stateful: each topic stores a value consisting of one or more records on the Diffusion server.

Update recordV2 topics on page 270
The following example demonstrates how to create and update recordV2 topics, including the use of a schema.

Subscribe to recordV2 topics on page 275
The following example demonstrates how to process information from subscribed recordV2 topics, including the use of a schema.

Related tasks

Defining a recordV2 schema on page 266
You can use the API to specify a schema that defines the content of a recordV2 topic.

Pub-sub

Having decided on your topic structure and the format of your data, consider how you publish the data through the topics.

Pub-sub is the primary model of data distribution used by Diffusion. Clients subscribe to a topic. When data is published to the topic as an update, the Diffusion server pushes that update out to all of the subscribed clients.

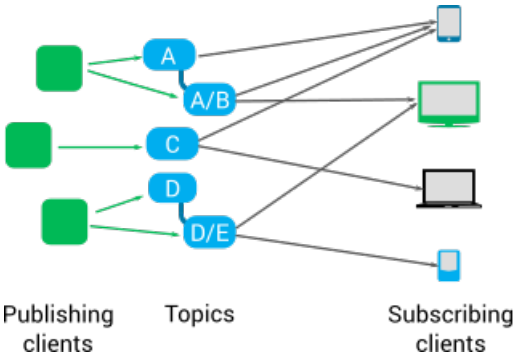


Figure 2: Pub-sub model

A client can both publish to topics and subscribe to topics, depending on the permissions that client has.

Concepts

Update

An update is data published to a topic by a client that is applied to the topic to change the topic state. The updated data is then pushed out to all subscribing clients.

State

The latest published values of all data items on the topic. The state of a topic is stored on the Diffusion server.

Value

A value is an update that contains the current state of all data on the topic.

Delta

A delta is an update that contains only those items of data that have changed on the topic since the last update was sent.

Topic loading

When a client first subscribes to a topic, it is sent a topic load message. A topic load is a value update that contains the current state of the topic.

Fetch

A request for the current state of all data on the topic. A client can fetch a topic's state without being subscribed to the topic. This request-response mechanism of getting data from a topic is separate from topic subscriptions.

Topic notifications

A client can register to receive topic notifications which provide information about which topics exist in the topic tree, but not the topic values. This is useful if your client needs to monitor the structure of the topic tree (or part of the tree) without the overhead of receiving all the values. Registering for notifications is separate from subscribing to a topic.

Publishing data

Consider the following information when deciding how to publish data to topics.

Data type

The updates that you publish to a topic must have a data type and format that matches the data type of the topic.

For example, if your topic is a single value topic where the data is of type integer, all updates published to the topic must contain a single piece of integer data.

Similarly, if your topic is a record topic with a metadata structure defined, all updates published to the topic must have the same metadata structure.

Updaters

You can use one of the following types of updater:

Value updater

This is the preferred type of updater to use with JSON and binary topics. When used as exclusive updaters, value updaters cache the values they use to update topics. This enables them to calculate and send deltas, reducing the volume of data sent to the Diffusion server.

Standard updater

This type of updater updates topics that use content to represent their data values. Updaters do not cache values and send all of the data passed to them to the Diffusion server without performing any optimization.

Both updater types can be used exclusively or non-exclusively.

For more information, see Updaters.

Exclusive updating

To update a topic exclusively, a client registers as the update source for that topic. Only one client can be the active update source for a topic and any attempts by other clients to update that topic fail.

Implementing exclusive updating is more complex than non-exclusive updating as it involves the extra step of registering as an update source.

A single client acting as the exclusive updater can be an advantage if you require that a single client has ownership of a topic or branch of the topic tree. This requires less coordination and management than updating a single topic from multiple clients.

If the ordering of the updates is important, use exclusive updating to ensure that a single client has control over what data is published and when.

If you are using high-availability topic replication, clients must update the replicated topics exclusively. Non-exclusive updates are not replicated by high-availability topic replication.

Non-exclusive updating

To update a topic non-exclusively, a client publishes updates to the topic and, if no other client has registered to update the topic exclusively, the update is applied to the topic.

Non-exclusive updating is the simpler way to update a topic.

Clients that update a topic non-exclusively risk their updates being overwritten by updates from other clients or that updates from multiple clients are published in a different order than intended.

If you use a value updater non-exclusively, the updater does not cache the value used to update the topic.

Non-exclusive updating is not supported with topics that are replicated using the high-availability capability.

Dynamically adding topics

A publishing client can create topics dynamically as and when the topics are required. For example, in response to a subscription request from another client for a non-existent topic.

Security

To publish data to a topic, a client must have the update_topic permission for that topic.

For more information, see Permissions on page 142.

Subscribing to topics

Consider the following information when deciding how clients subscribe to topics.

For a client to receive a stream of updates from a topic, the following conditions must be met:

- The client must subscribe using a selector that matches the topic.
- The topic must exist on the Diffusion server.
- The client must register at least one stream that matches the topic.

When all these conditions are met, the stream receives a subscription notification and an initial value for the topic. The client receives subsequent updates to the topic through the stream.

The order in which these conditions are met does not affect the receipt of the subscription notification and the initial value of the topic.

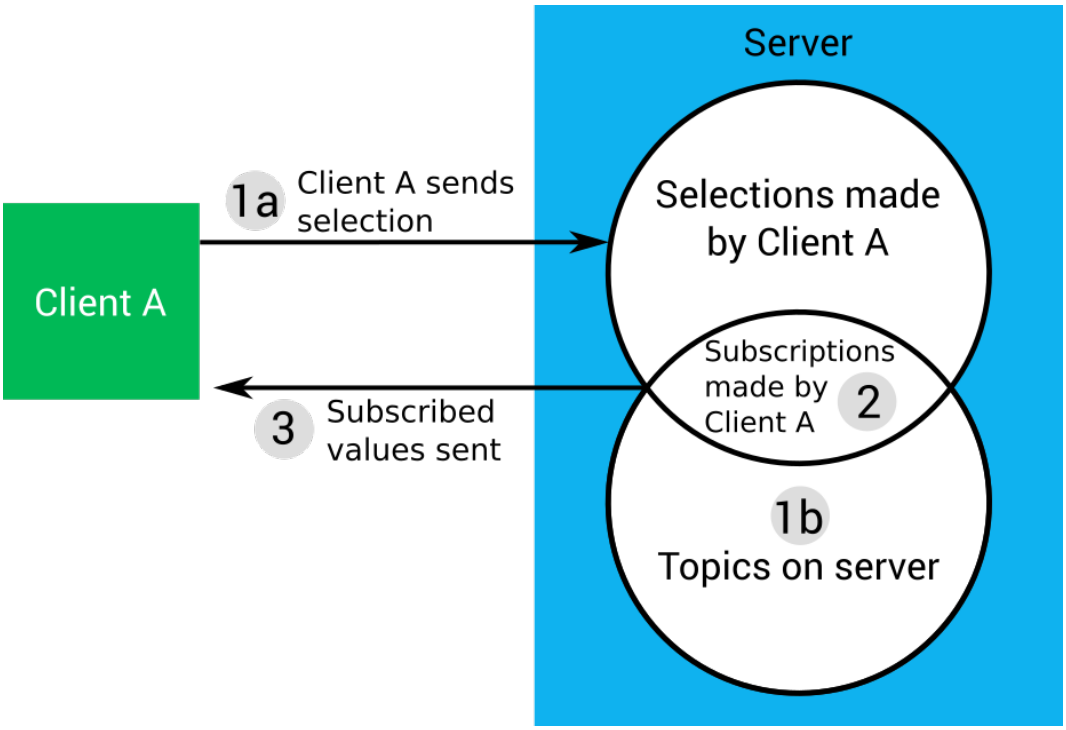
An exception is in the case of record topics. For more information, see Considerations when using legacy record topics on page 98.

Permissions

To subscribe to a topic, a client must have the select_topic permission and the read_topic permission for that topic. For more information, see Permissions on page 142.

The rest of this section assumes that the client has the required permissions to complete the described actions.

Subscription flow



1. The prerequisite conditions for subscribing to a topic are met:

a. The client selects a set of topics to subscribe to, or that selection is made on the client's behalf by another client.

   A client can select multiple topics using a topic selector. This subscription can be to topics that match a particular regular expression or to topics in a particular branch of the topic tree. For more information, see Topic selectors on page 44.

   The selection made by the subscribe request is persistent and is stored on the Diffusion server. Because selections are stored, the client can subscribe, pre-emptively, to topics that do not currently exist.

b. The topic exists on the Diffusion server.

   The topic can be created before or after any subscribe request that selects it. In both cases, the client that makes the request is subscribed to the topic when both the selection and the topic exist.

2. Both prerequisite conditions for a subscription are met and the client is subscribed to the topic.

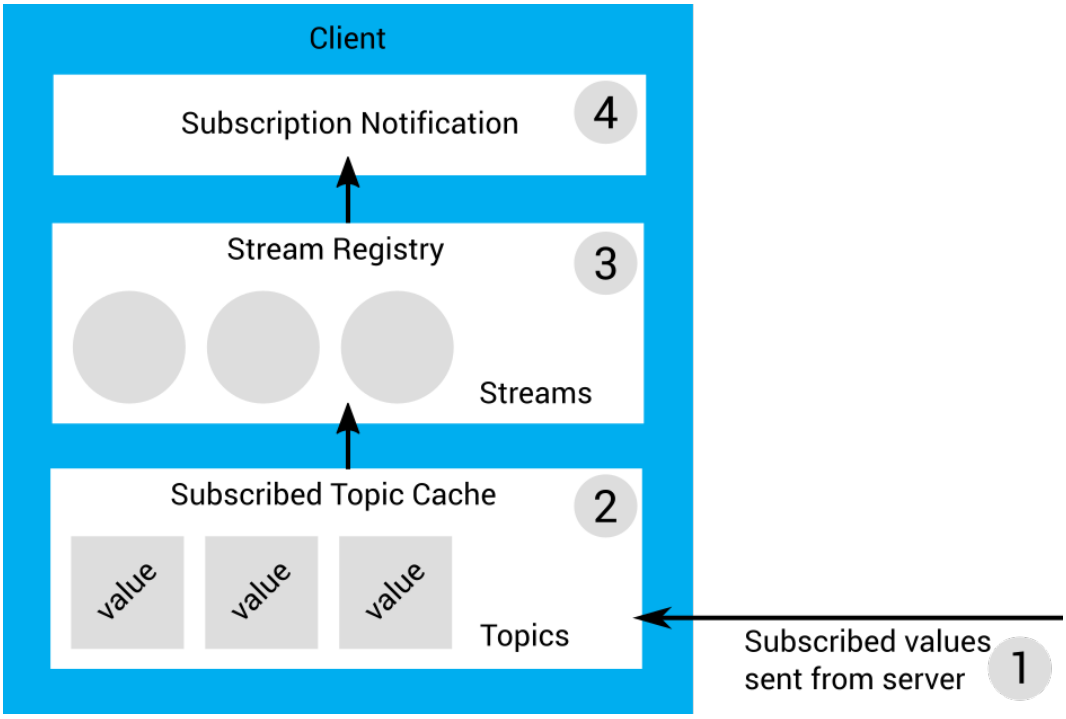
   The intersection of the topic paths that the client has selected for subscription and the topics that exist on the Diffusion server defines the list of subscriptions that client has made.

   For each client that connects to the Diffusion server, the Diffusion server stores a separate list of subscriptions.

3. When the subscription is made, the value of the subscribed topic is sent to the client.

   Subsequent updates are sent as values or as deltas, depending on the topic specification and the nature of the update.

Client handling of data from subscribed topics



1. When the client makes a subscription to a topic, the value of the topic is sent to the client.

   Subsequent updates are sent as values or as deltas.

2. Values for each subscribed topic are stored in the subscribed topic cache on the client.

   This is not the case for stateless or record topics, see Considerations when subscribing on page 98 and Considerations when using legacy record topics on page 98.

3. The client registers streams against a topic selector. A list of streams is maintained in the stream registry.
4. If one or more streams exists that matches a topic, the value for that topic is received through the matching streams.

Considerations when subscribing

The subscriptions a client has, which are defined by the intersection of the topics that exist on the Diffusion server with the selections made by the client, determine what data is sent to the client from the Diffusion server. To reduce the amount of data sent between the Diffusion server and the client, only subscribe to those topics that the client uses.

The streams a client registers determine what topic values are available for the client to work with. Adding and removing streams as they are needed by the client enables your application to access topic values stored in the subscribed topic cache in real time.

When a client subscribes to a stateless topic, the values received are not stored in the subscribed topic cache. If a stream is created after the topic is subscribed, the stream does not receive a value until the next time the topic is updated.

Considerations when using legacy record topics

When a client subscribes to a topic that uses the deprecated record type, the values received are not stored in the subscribed topic cache. If a stream is created after the topic is subscribed, the stream might only receive delta values. In this case, the client has no value to apply the deltas to and the data is incorrect. When subscribing to record topics, always create the stream before requesting a subscription.

This does not apply to the recordV2 topic type.

Topic notifications

Topic notifications enable a client to receive information about the topic tree structure, without topic values.

A client can receive notifications about changes to selected topics through the topic notifications feature.

The client must use a topic notification listener to receive notifications. Use topic selectors to specify which topics the client will be notified about.

Selection and deselection

A client can request selections at any time, even if the topics do not exist at the server. Selections are stored on the server and any subsequently added topics that match registered selectors will generate notifications.

Notification contents

Each notification includes:

- The topic specification of the topic
- A notification type describing the change

Table 16: Notification types

Value	Meaning
ADDED	A new topic has been added matching a registered selector
REMOVED	A selected topic has been removed
SELECTED	A newly-registered selector matched a topic that already exists
DESELECTED	An existing topic is no longer selected due to a selector being removed

For example, suppose a topic tree contains only the topic a/b/c. A listener registers the topic selector ? a// which selects the topic a and all topics below it.

The listener will receive a notification containing the topic specification of the topic at a/b/c, and the notification type SELECTED.

If a new topic is added at a/b/c/d, another notification will be received with the specification of the new topic, and a notification type ADDED.

Immediate descendant notifications

Listeners receive notifications about whether each selected topic has unselected immediate descendants.

An immediate descendant means the first bound topic on any branch below a given topic path. By monitoring immediate descendant notifications, you can implement a listener which selects deeper topic paths as more topics are added, in order to walk the topic tree.

For example, in a topic tree which contains only these topics:

- a
- a/b
- a/c
- a/c/d
- a/e/f/g

The immediate descendants of a are a/b, a/c and a/e/f/g.

a/c/d is not an immediate descendant of a, because its parent a/c is a descendant of a.

Immediate descendant notifications provide the topic path and a notification type (with the same possible values as above).

In the example topic tree above, suppose that a topic notification listener had selected topic a using the topic selector "a". If a topic is now added at a/x, the listener receives an immediate descendant notification with the path a/x and the notification type ADDED.

If a topic is then added at a/x/y, the listener does not receive another notification, because a/x/y is not an immediate descendant.

Considerations when using topic notifications

Topic notifications are useful when your client needs to know which topics are present, without the overhead of receiving the topic values. This can be useful when developing monitoring tools or interfaces designed to browse large numbers of topics.

A client will only be notified about topics for which it has both select_topic and read_topic permissions.

The select_topic permission is required to select a topic with a listener. The read_topic permission is required to receive notifications for a topic.

Request-response messaging

You can send request messages directly to a client session, a set of client sessions, or a message path. The recipient of a message can respond to the request.

Concepts

Request

A message sent from one client session to another session, to a message path, or to a set of sessions.

Response

A message sent in reply to a request message.

Data type

Request and response messages can contain data of one of the following types: JSON, binary, string, 64-bit integer, or double.

The response message is not required to be the same data type as the request it responds to.

Message path

The path used to address the request messages.

The message path is made up of path segments separated by the slash character (/). Each path segment can be made up of one or more Unicode characters.

Handler

An object registered by client session to handle requests sent on message paths in a specific branch of the path hierarchy, and to respond to those requests.

Stream

An object used by a client session to receive requests sent to that client session, and to respond to those requests.

Session properties

Properties assigned to a session, either by the Diffusion server or by an authentication handler. These properties can be used to select the set of sessions to send requests to.

For more information, see Session properties on page 220.

Send request messages in the following ways:

Send requests to a message path

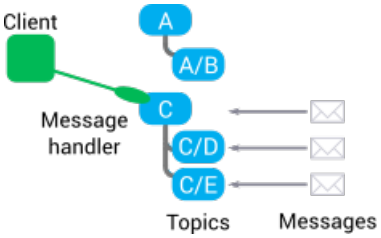


Figure 3: A client session registers a handler on part of the topic tree

A client session with the send_to_message_handler permission can send requests on a message path. The sending client session does not know which client session, if any, receives the request.

A client session with the register_handler permission can register a handler on a part of the topic tree. This client session receives any requests that are sent on message paths in that part of the topic tree and sends a response.

For more information, see Sending request messages to a message path on page 325.

Send request messages to a specific client session

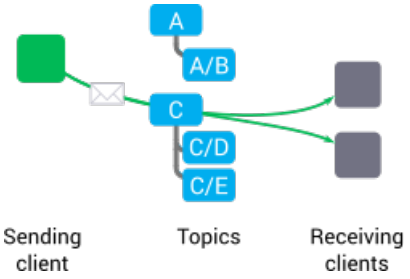


Figure 4: A client session can send requests through a message path to a known client session

A client session with the send_to_session permission that knows the session ID of a client session can send requests through a message path to the known client session.

The responding client must have a request stream registered against a message path to receive requests sent through that message path and respond to them.

For more information, see Sending request messages to a session on page 331.

Send request messages to a set of client sessions

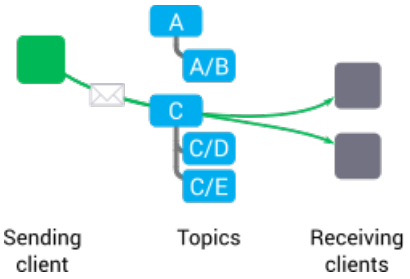


Figure 5: A client can send requests through a message path to a set of client sessions

A client session with the send_to_session permission can send requests through a message path to a filter that selects client sessions based on their session properties.

The responding client session must have a request stream registered against a message path to receive and respond to requests sent through that message path.

For more information, see Sending request messages to a session filter on page 338.

Considerations when using request-response messaging

- The data type of the request is not required to match the data type of the response. For more information, see Typed requests and responses on page 323.
- Messaging can use message paths that are the same as topic paths with topics bound to them. However, there is no connection between messaging and topics. For more information, see Message path on page 324.
- Request-response messaging is cluster-aware. This means that a message handler attached to one server in a Diffusion cluster can route a message to any server within that cluster.

Conflation

Conflation of messages is the facility to reduce the amount of information sent to clients by combining or discarding updates.

The server has a separate outbound queue for each client session.

Using conflation, the server examines the outbound queue and removes or combines updates which are stale or redundant.

Conflation is an optional feature that can be applied selectively to certain topics or client sessions.

Advantages of message conflation

Conflation can reduce the server memory footprint as well as the amount of network data transmitted.

It can also prevent sessions being closed due to the maximum queue size limit being exceeded.

Considerations when using conflation

- Do not use conflation if there are relationships or dependencies between topics. Conflation alters the order of updates. If a conflated topic is temporally or causally related to another topic, conflation can cause unwanted behavior.
- Do not use conflation if individual updates carry forensic storage or audit trail requirements.

Related concepts

Configuring conflation on page 385
Use the `CONFLATION` topic property to select a conflation policy for a topic.

Using conflation

You can configure how and when conflation is applied to different topics.

Conflation policies

Conflation policies control how conflation is applied to a topic. You can set conflation policy for a topic with a topic property.

These are the available conflation policies:

- off
- conflate (default)
- unsubscribe
- always

"off" disables all conflation for the topic. Topic updates will never be merged or discarded.

"conflate" automatically conflates topic updates when back pressure is detected by the server (that is, when the outgoing message queue exceeds the maximum allowed size in bytes or number of messages).

"unsubscribe" automatically unsubscribes the topic when back pressure is detected by the server, with a BACK_PRESSURE message. The unsubscription is not persisted to the cluster, so if a session fails over to a different server it will be resubscribed to the topic. This policy is useful for topics that are not essential to the application, and can be discarded in back pressure situations without affecting the main function of the application.

"always" automatically conflates topic updates as they are queued for the session. This policy ensures only the latest update is queued for the topic, minimising the server memory and network bandwidth used by the session, but potentially increasing the processor cost of conflation.

If no policy is set, the "conflate" policy is applied.

Conflation process

The conflation process considers the value and delta updates in the queue, and the current topic value (unless DONT_RETAIN_LAST_VALUE is enabled). It reduces the queued updates to a single value or a composite delta, whichever requires the fewest bytes to send.

Under the default "conflate" policy, no conflation is applied until there is a new message to send to a session with a full queue. The whole queue is then conflated, topic by topic. If conflation is not enough to bring the queue size under the configured limit, the server will close the session.

Related concepts

Configuring conflation on page 385
Use the `CONFLATION` topic property to select a conflation policy for a topic.

Designing your solution

Decide how your solution components interact to most efficiently and securely distribute your data.

There are a number of things to consider when designing your Diffusion solution:

- The number, distribution, and configuration of your Diffusion servers
- How you use clients in your solution
- The additional components to develop
- The third-party components you might include in your solution
- Securing your solution

These considerations are not separate. The decisions you make about one aspect of your solution can affect other aspects.

Scaling out your solution

In production scenarios, it is likely that your Diffusion solution will need to scale out. Fan-out is the preferred choice for scaling out. However, it is also good practice to cluster the primary tier of a fan-out solution.

Servers

Consider the quantity, distribution, location and configuration of your Diffusion servers.

How many Diffusion servers?

Consider the following factors when deciding how many Diffusion servers to use in your solution:

Number of client connections

How many client connections do you expect to occur concurrently? For a greater number of concurrent client connections, you might require more Diffusion servers to spread the load between.

Volume of data

At what rate are you publishing updates and sending messages? How large are the updates and messages? If you are distributing a greater volume of data, you might require more Diffusion servers to spread the load between.

Hardware capabilities

The number of concurrent client connections and the volume of data that a single Diffusion server can handle depend on the hardware that the Diffusion server runs on.

In order of importance, the following hardware components have the biggest impact on the server performance:

- Network interface controller (NIC)
- Central processing unit (CPU)
- Random access memory (RAM)

Resilience and failover requirements

Ensure that you have enough Diffusion servers that if one or more becomes unavailable, for example when updating the server or due to a failure of the hosting system, the remaining Diffusion servers can spread the resulting load increase.

You can also use replication between Diffusion servers to increase your solution's resilience. For more information, see Server clusters for high availability on page 105.

Distribution of servers

How you wish to distribute your servers has an effect on how many servers you require.

For example, if your client base is distributed geographically, you might want to locate your Diffusion servers in different territories. This enables your servers to be more responsive because of their proximity to clients. In this case, the number of territories your client base is spread over affects the number of servers you require.

You can easily scale your solution by adding additional Diffusion servers if your requirements change.

How are your Diffusion servers configured?

Consider the following factors when deciding how to configure the Diffusion servers in your solution:

Ports

What ports do you want to provide access to your Diffusion server on? By default, your Diffusion server supports client connections on port 8080.

Reconnection behavior

Do you want to allow clients that lose their connection to reconnect to the server? How long do you want to keep client sessions available after the client loses connection?

Clusters

Replication enables a cluster of Diffusion servers to share information about configuration, topics and client sessions with each other through a data grid, providing redundancy and high availability .

For more information, see Server clusters for high availability on page 105.

Data sharing with remote servers or between clusters

The remote topic view and fan-out features enable data to be shared between servers that are not in a cluster. This enables you to create local secondary servers for load-balancing, or share data with a geographically remote server cluster to minimise latency.

For more information, see Sharing data with remote servers on page 113.

Performance

Tuning your Diffusion servers for performance is best done as part of testing your solution before going to production. This enables you to observe the behavior of your solution in action and configure its performance accordingly.

For more information, see Tuning on page 497.

For more information, see Configuring your Diffusion server on page 414.

This manual describes the factors that you must consider when designing your Diffusion solution. However, these factors are too many and too interlinked for this manual to provide specific guidance.

Push Technology provides Consulting Services that can work with you to advise on a solution that best fits your requirements. Email for more information.

Server clusters for high availability

Consider how to replicate session, topic and configuration information between a cluster of Diffusion servers to increase availability and reliability.

Diffusion uses a datagrid to share session and topic information between Diffusion servers within a cluster, providing high availability for clients connecting to load-balanced servers.

Figure 6: Information sharing using a datagrid

Diffusion uses Hazelcast™ as its datagrid. Hazelcast is a third-party product that is included in the Diffusion server installation and runs within the Diffusion server process.

The datagrid is responsible for the formation of clusters and the exchange of replicated data. These clusters operate on a peer-to-peer basis and by default there is no hierarchy of servers within the cluster.

Servers reflect session and topic information into the datagrid. If a server becomes unavailable, another server can access the session and topic information that is stored in the datagrid and take over the responsibilities of the first server.

As well as session and topic information, servers can use configuration replication to replicate configuration items such as security stores, topic views and metric collectors.

Configuration replication is active if session or topic replication is enabled, or it can be enabled separately.

Many Diffusion features are cluster-aware, meaning that requests or messages can be routed within a cluster to the correct server. These features are cluster-aware:

- control authentication handler requests
- missing topic notifications
- request-response messaging

Some client control operations are cluster-aware. The command will be routed to the server in the cluster that hosts the specified session. When sending a request to a session filter, the command is applied to all matching sessions across the cluster.

These client control operations are cluster-aware:

- changeRoles
- close
- setConflated
- setSessionProperties
- getSessionProperties

See Configuring the Diffusion server to use replication on page 466 and Replication.xml on page 469 for more details.

Considerations

Consider the following factors when using replication with Hazelcast:

- By default Hazelcast uses multicast to discover other nodes to replicate data to. This is not secure for production use. In production, configure your Hazelcast nodes to replicate data only with explicitly defined nodes. For more information, see Configuring the Hazelcast datagrid on page 467.
- When Diffusion servers are merged into a cluster, the servers can have inconsistent replicated data. Unresolved inconsistencies can cause unpredictable behavior, due to issues such as conflicts between updaters. If the inconsistencies cannot be resolved, this is known as "split-brain". The inconsistent Diffusion server or servers are shutdown and must be restarted.

  Diffusion servers in a cluster can become inconsistent in a number of circumstances; for example, if a network partitions and then heals.

  The quorum setting can help prevent inconsistencies due to network partitions. It enables you to set a minimum size for a cluster, below which the servers in a cluster will all shut down.

  You should choose a quorum value so that after a network partition, the smaller cluster will shut down instead of attempting to heal. The servers from the smaller cluster can then be restarted and join the cluster cleanly, avoiding inconsistencies.

  If you want to use the quorum feature, use an odd number of servers and set the value to just over half the cluster size. For example, if you have 5 servers in a cluster, set the quorum value to 3.

Note that servers shut down by the quorum feature will not restart automatically.

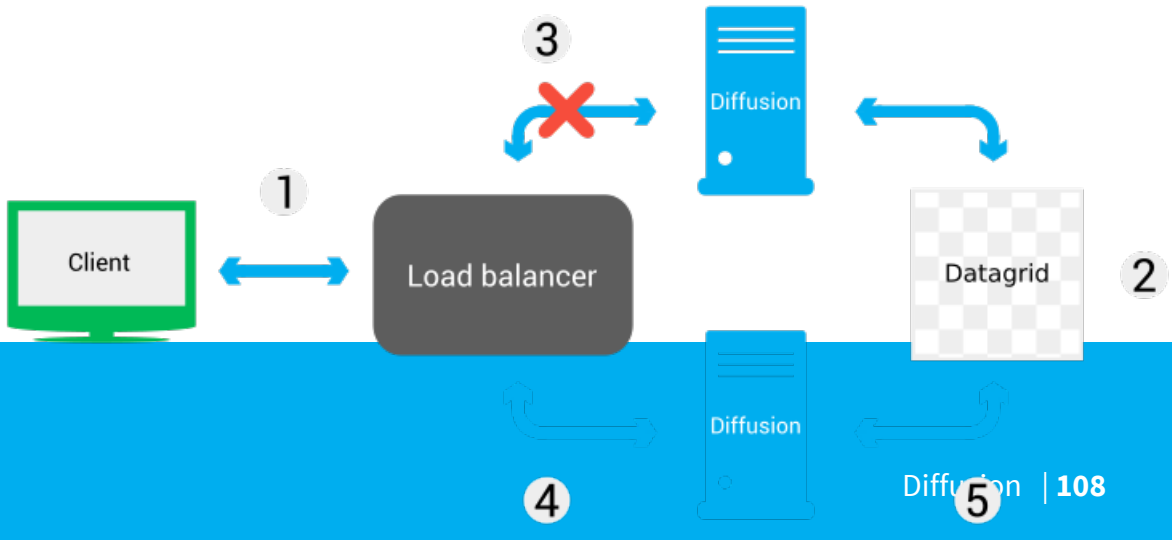
- An ideally sized cluster contains at least 3 nodes, and no more than 5 without consultation. Design your cluster to contain an odd number of servers, as these cannot fail to recover from a "split-brain".

Session replication

You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

When a connection from a client through the load balancer to a Diffusion server fails, the load balancer routes the client connection to another Diffusion server. This server has access to the session and client information that is replicated in the datagrid.

Clients that connect to a specific Diffusion server and not through a load balancer cannot use session replication.


3
Diffusion
1
Client
Load balancer
Datagrid
2
Diffusion
4
5

The load balancer is configured to route based on the client's session ID and requests from the client go to the same server until that server becomes unavailable.

2. Information about the client session is reflected into the datagrid.

   The following information is replicated:

   - session ID
   - session principal
   - session properties
   - list of topic selections

   The following information is not replicated and is created anew on the server a client fails over to:

   - session start time
   - statistics
   - client queue
3. A client loses connection to the Diffusion server if the server becomes unavailable.
4. The client can reconnect and the load balancer routes the connection to another Diffusion server.
5. This Diffusion server has access to all of the client information shared into the datagrid by the first Diffusion server.
6. The server uses the list of topic selections to recover the set of subscribed topics and subscribes the client to these topics.
7. Subscribing the client to topics provides full value messages for all topics that contain the current topic state.

The client can reconnect to its session only if it reconnects within the reconnect time specified in the `Connectors.xml` configuration file. If the client does not reconnect within that time, the client session information is removed from the datagrid.

Considerations

Consider the following factors when using session replication:

- Replication of session information into the datagrid is not automatic. It must be configured at the server.
- Messages in transit are not preserved.
- When a client session reconnects it does not need to authenticate again. The client uses a session token to reacquire its session. Ensure that this token is secure by using a secure transport to connect, for example, WSS.
- The failover appears to the client as a disconnection and subsequent reconnection. To take advantage of high server availability, clients must implement a reconnect process.
- The Diffusion server that a client reconnection attempt is forwarded to depends on your load balancer configuration. Sticky load balancing can be turned on to take advantage of reconnection or turned off to rely on session replication and failover.

Differences between session reconnection and session failover

When a client loses a load-balanced connection to Diffusion, one of the following things can occur when the client attempts to reconnect through the load balancer:

Session reconnection

The load balancer forwards the client connection to the Diffusion server it was previously connected to, if that server is still available. For more information, see Reconnect to the Diffusion server on page 203.

Session failover

The load balancer forwards the client connection to a different Diffusion server that shares information about the client's session, if session replication is enabled between the servers.

Prefer session reconnection to session failover wherever possible by ensuring that the load balancer is configured to route all connections from a specific client to the same server if that server is available.

Session reconnection is more efficient as less data must be sent to the client and has less risk of data loss, as sent messages can be recovered, in-flight requests are not lost, and handlers do not need to be registered again.

For more information, see Routing strategies at your load balancer on page 626.

To a client the process of disconnection and subsequent reconnection has the following differences for session reconnection or session failover.

Session reconnection	Session failover
The client connects to the same Diffusion server it was previously connected to.	The client connects to a Diffusion server different to the one it was previously connected to.
The client sends its last session token to the server.	
The server authenticates the client connection or validates its session token.	
The server uses the session token to resynchronize the streams of messages between the server and client by resending any messages that were lost in transmission from a buffer of sent messages. If lost messages cannot be recovered because they are no longer present in a buffer, the server aborts the reconnection.	The server uses the session token to retrieve the session state and topic selections from the datagrid.
The server sends any messages that have been queued since the session disconnected.	The server uses the state to recover the session, uses the topic selections to match the subscribed topics, and sends the session the current topic value for each subscribed topic. Any in-flight requests made by the client session to the previous server are cancelled and the client session is notified by a callback. All handlers, including authentication handlers and update sources, that the client session had registered with the previous server are closed and receive a callback to notify them of the closure.

Related concepts

Session reconnection on page 505
You can configure the session reconnection feature by configuring the connectors at the Diffusion server to keep the client session in a disconnected state for a period before closing the session.

Related tasks

Configuring the Diffusion server to use replication on page 466
You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

Related reference

Topic replication on page 111

You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Configuring the Hazelcast datagrid on page 467
You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Replication.xml on page 469
This file specifies the schema for the replication properties.

Topic replication

You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

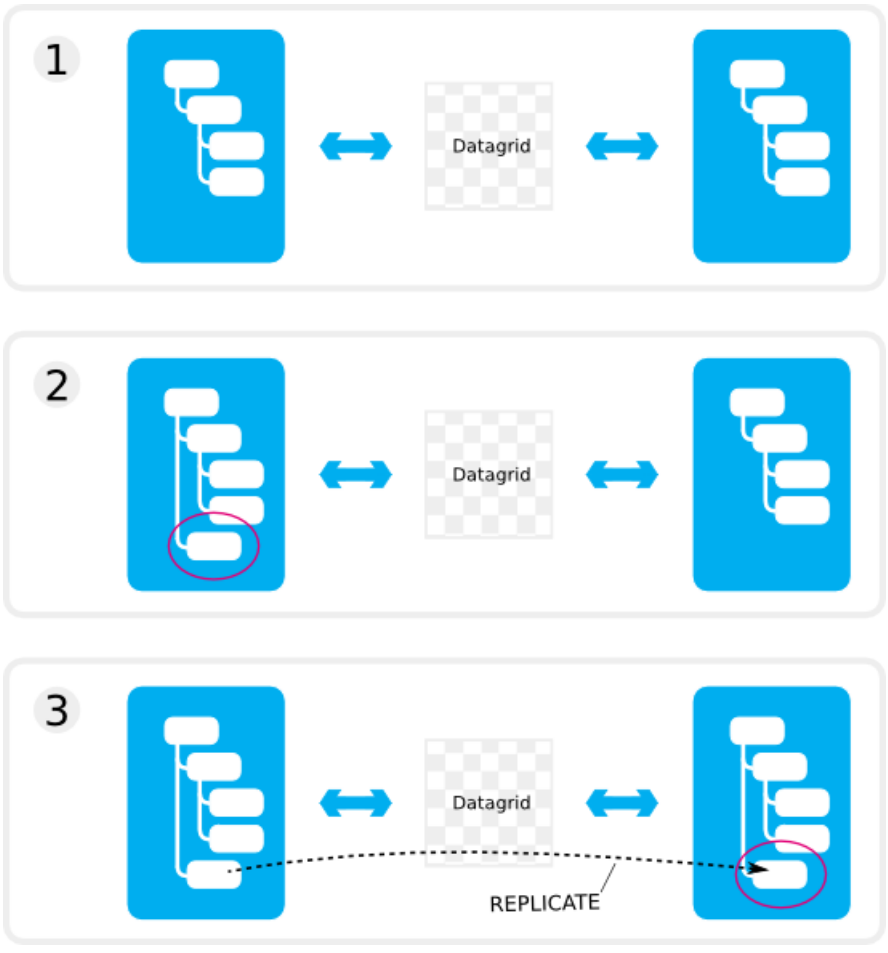


Figure 8: Topic replication

1. Servers with topic replication enabled for a section of the topic tree share information about that section of the topic tree through the datagrid. The topic information and topic data are synchronized on all the servers.
2. A new topic is created on one server in the replicated section of the topic tree.

3. The new topic is replicated on the other servers with identical topic information. When its topic data is updated on the first server, that data is replicated on the other servers.

Considerations

Consider the following factors when using topic replication:

- Avoid registering a large number of update sources. Do not design your solution so that each topic has its own exclusive update source. This will cause performance problems when starting new servers and joining them to existing clusters, due to the overhead of sharing the update source registrations.
- Only publishing topics can be replicated.
- Replication is not supported for routing topics.
- Any topic that is part of a replicated branch of the topic tree and is not one of the supported types of topic is not replicated. Instead that topic path remains unbound.
- Only topic-wide messages are replicated. Messages sent to a single client or to all clients except one are not replicated.
- Replication of topic information into the datagrid is not automatic. It must be configured at the server. This gives a performance advantage, as you can choose which parts of your topic tree to replicate.
- Updates to a particular topic will always be delivered to all subscribers in the same order. However, delivery order is not guaranteed across different topics. For example, if you update topic 1 then topic 2, some subscribers might receive the update to topic 2 before the update to topic 1.
- Replication of topic data can impact performance.

Related tasks

Configuring the Diffusion server to use replication on page 466
You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

Related reference

Session replication on page 107
You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Configuring the Hazelcast datagrid on page 467
You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Replication.xml on page 469
This file specifies the schema for the replication properties.

Failover of active update sources

You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

A client must register as an update source to update a replicated topic. Replicated topics cannot be updated non-exclusively. For more information about update sources, see Updating topics on page 291.

1. A client (CLIENT 1) connects to a Diffusion server (SERVER 1) and registers an update source for a section of the topic tree that has topic replication enabled. This update source is the active update source.
2. Another client (CLIENT 2) connects to another Diffusion server (SERVER 2) and registers an update source for the same section of the topic tree. This update source is a standby update source.
3. The topics on SERVER 2 continue to receive their updates from CLIENT 1 through the datagrid.
4. If SERVER 1 or CLIENT 1becomes unavailable, the update source registered by CLIENT 2 becomes active. SERVER 2 sends CLIENT 2 a callback to notify it that it is the active update source.

   On SERVER 2, the topics in that section of the topic tree receive their updates from CLIENT 2. SERVER 2 reflects this topic data into the datagrid.

Considerations

Consider the following factors when using failover of active update sources:

- If the topic paths that the updating client uses to register an update source do not match the topic paths configured in the `Replication.xml` configuration file of the server, unexpected behavior can occur.
- The mechanism that provides failover of active update sources assumes that all servers have the same configuration and that all control clients implement the same behavior as part of a scalable and highly available deployment. If this is not the case, unexpected behavior can occur.

Related concepts

Updating topics on page 291
A client can use the TopicUpdate feature to update topics.

Related tasks

Configuring the Diffusion server to use replication on page 466
You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

Related reference

Session replication on page 107
You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

Topic replication on page 111
You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

Configuring the Hazelcast datagrid on page 467
You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Replication.xml on page 469
This file specifies the schema for the replication properties.

Sharing data with remote servers

Remote topic views enable you to share topic data between servers that are not in the same cluster.

Remote servers versus server clusters

Diffusion can replicate topics across a cluster for high availability, but for some use cases, you need to share topic data between servers which are not in the same cluster.

For example:

- You need to feed the same topic data to servers in multiple geographic regions, to minimise latency for end users
- You need to share a subset of your topic data to a Diffusion server (or cluster) administered by a separate organisation
- You need separate topic data-hosting and session-hosting clusters, allowing the session-hosting cluster to be scaled up to support more connections.
- You need clusters separated by a firewall to share topic data.

We refer to Diffusion servers that are not part of a cluster as "remote servers".

> **Note:** A remote server in this sense is not always physically remote: for some applications it could be in the same data center, or even on the same LAN. The important distinction is that it is not part of the same Diffusion cluster. Diffusion will efficiently transfer topic data with minimum latency and bandwidth usage even if the server is geographically remote or the network is constrained.

Remote topic views versus fan-out

Diffusion offers two ways to share topic data between servers that are not in the same cluster:

Remote topic views

Remote topic views were introduced in Diffusion 6.5. A remote topic view uses the existing topic view mechanism to map part of the topic tree from a remote server.

A remote topic view can be combined with the other capabilities of topic views, enabling sophisticated dynamic mapping and transformation of topics, as well as delayed or throttled feeds.

Remote topic views are configured dynamically with the API or management console and the configuration is automatically shared within a cluster, whereas fan-out requires file-based configuration.

In previous versions, missing topic notifications could not propagate through a remote topic view. As of Diffusion 6.7, missing topic notifications can propagate through a remote topic view.

Fan-out

Fan-out is an older mechanism used to replicate topics from a remote (or primary) server to a secondary server where the fan-out link is configured.

There is no longer any reason to use fan-out, and it is now deprecated.

For a new application, you should use remote topic views in preference to fan-out.

If your existing application still uses fan-out, you should update it to use remote topic views instead. Fan-out support will be removed in a future release.

Related concepts

Creating a remote server definition on page 396

To connect to a Diffusion server that is not in the same cluster, create a remote server definition. This can be used by a remote topic view.

Remote topic views

Consider whether to use remote topic views to share topic data between servers that are not part of the same cluster.

Remote topic views are an extension of topic views. A topic view maps one part of the topic tree to another part. Usually this takes place within a server, but a remote topic view can map topics from a remote server.

Why use a remote topic view?

A remote topic view is useful if you need to make the same data available on multiple servers where clustering is not suitable. This might be for geographical reasons to minimise latency, or because the servers are separated by a firewall.

You can also use the other topic view transformations to process topic data from the remote server and create derived topics (for example, transforming topic values and tree structure or delaying or throttling updates).

How remote topic views work

Like any topic view, a remote topic view creates reference topics based on a set of source topics.

A remote topic view is configured on the server (or cluster) that will receive the reference topics, derived from source topics on the remote server.

The remote topic view is defined with a topic view specification and a remote server definition, both of which can be created with a client API. See Creating a remote server definition on page 396 for details.

Automatic topic removal with remote topic views

Automatic topic removal can monitor subscriptions to a reference topic created from a normal topic on a remote server with a topic view.

However, subscriptions cannot be monitored if a reference topic is created from a reference topic. For example, suppose a normal topic view is used to create a reference topic on a remote server. If a remote topic view is used to replicate that reference topic, subscriptions to the resulting reference topic will not be counted for automatic removal purposes.

If you want to combine automatic topic removal and remote topic views, avoid using topic views on the remote server to create any topics that could be selected by the remote topic view.

Routing topic limitation

Remote topic views have a limitation when using the deprecated routing topic type.

A routing topic will only resolve to a reference topic if the routing handler at the remote server resolves a value for the remote server client.

Creating a remote topic view

For details of how to create a remote topic view, see Remote topic views on page 65.

Fan-out

Fan-out is a way to replicate topic information from primary servers on one or more secondary servers.

A fan-out distribution comprises many servers that host the same topic or topics. When the topic is updated on a primary server, the update is fanned out to replica topics on secondary servers.

> **Note:** Remote topic views are generally a superior alternative to fan-out. They are easier to configure and can also be combined with other topic view capabilities to transform data. Fan-out, however can be used if all that is required is simple one to one mapping of topics from primary to secondary servers as it has a smaller memory footprint.

How fan-out works

Fan-out is configured on the secondary server or secondary servers in the solution.

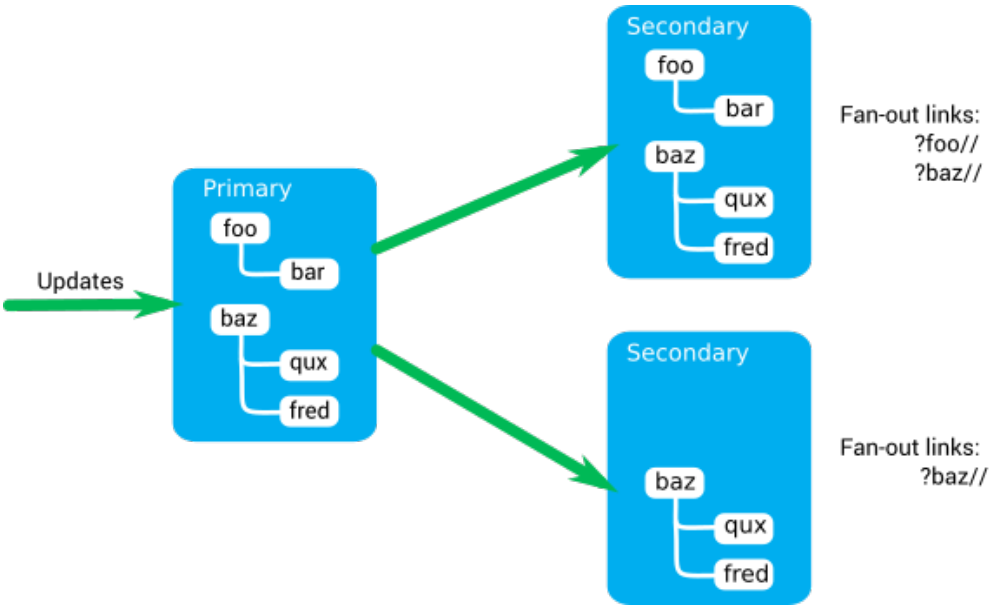


Figure 9: Fan-out

- A secondary server connects to a primary server as a client.
- The secondary server subscribes to a set of topics on the primary server.

  This set of topics is defined by a selector in the configuration of the secondary server.
- The secondary server replicates the subscribed topics locally.
- When updates are made to the topics on the primary server, the secondary server receives these updates through the standard pub-sub feature in the same way as any other client of the primary server.
- The secondary server applies the updates to its replica topics.
- Any clients subscribed to a replica topic on the secondary server receive the updates through the standard pub-sub feature.
- If a topic is removed at the primary server, the secondary server removes its replica topic.
- If a topic is added at the primary server that matches the set of topics subscribed by the secondary server, the secondary server creates a local replica topic.

A secondary server can connect as a client and subscribe to topics on more than one primary server. However, ensure that the secondary server does not attempt to replicate the same topic from multiple sources as this can cause the data on the topic to be incorrect.

Whether a server is "primary" or "secondary" is a property of a particular fan-out link. You can create a configuration where a server is primary for one fan-out link, but secondary for another.

Creating topics on the primary server is an asynchronous action, because of this a client that creates a topic on the primary server receives a completed callback saying that the topic has been created. However, receiving this callback does not indicate that the topic has been replicated by fan-out and created on a secondary server.

Topic aliasing is not supported for topics that are replicated by fan-out. Ensure that aliasing is not enabled at the primary server.

Routing topics and fan-out

To use fan-out with routing topics, the routing subscription handlers for a routing topic must be registered at all secondary servers, but not at the primary server.

Slave topics and fan-out

A primary server will only replicate a slave topic to a fan-out secondary server if the topic is bound to a master topic that also exists on on the primary.

If the master topic is removed from the primary, both the master and the slave will be removed from the secondary.

Topic replication and fan-out

A secondary server cannot replicate the same topic from more than one primary server or multiple times from the same primary server. Validation of the path prefix of the selectors is in place to prevent this occurring, but the use of regular expressions in topic selectors can result in an overlap of replication which can cause problems.

Missing topic notifications generated by subscription or fetch requests to a secondary server are propagated to missing topic handlers registered against the primary servers. For more information, see Using missing topic notifications with fan-out on page 118.

Fan-out readiness

If you add a secondary server to a load balancer pool before all topics have propagated from the primary server, it can result in a large number of messages being generated, leading to `MESSAGE_QUEUE_LIMIT_REACHED` errors appearing in the logs.

As of Diffusion 6.4, you can avoid this problem using the fan-out readiness start condition configured in `etc/Connectors.xml`. If you enable fan-out readiness, the connector will not be started until a fanout link with specified connection and link names exists, has connected to the primary server, and has synchronized the topics from the primary server that match its configuration.

Reconnection and disconnection

You can configure fan-out servers to use the standard reconnect mechanism. If the connection between the secondary server and the primary server is lost, the secondary server can reconnect to the same session. However, if messages are lost between the primary and secondary server, the reconnection is aborted and the session closed. The secondary server must connect again to the primary server with a new session.

If a disconnection between the primary and secondary server results in the session being closed, the secondary server removes all the topics that it has replicated from that primary server. (Only topics explicitly defined by a selector are removed.) Clients subscribing to these topics on the secondary server become unsubscribed. If the secondary server connects again to that primary server with a new session, the secondary server recreates the topics. Clients connecting to the secondary server become resubscribed to the topics.

Related concepts

Configuring fan-out on page 418
Configure the the Diffusion server to act as a client to one or more other Diffusion servers and replicate topics from those servers.

Using missing topic notifications with fan-out

Missing topic notifications generated by subscription requests to a secondary server are propagated to missing topic handlers registered against the primary servers.

Control client sessions can use missing topic notifications to monitor the activity of end-user client sessions. In response to subscription requests to missing topics, the control client session can choose to take an action, such as creating the missing topic.

For more information, see Handling subscriptions to missing topics on page 254.

How notification propagation works

A missing topic notification is propagated from a secondary server to a missing topic handler registered against a primary server if and only if all of the following conditions are met:

- There is an active fan-out connection between the secondary server and the primary server.
- The selector used for the subscription request to the secondary server intersects with one or more of the fan-out links to the primary server that are configured at the secondary server.
- On the secondary server, there are no currently replicated topics that match both the fan-out link and selector used in the subscription request.
- The primary server has no topics that match the selector used in the subscription request.
- One or more missing topic handlers are registered against the primary server for a path that matches the selector. The following rules are used to select which missing topic handler receives the notification:
  - If multiple handlers are registered for the branch, the handler for the most specific topic path is notified.
  - If there is more than one handler for a path, the Diffusion server notifies a single handler.

Example flow

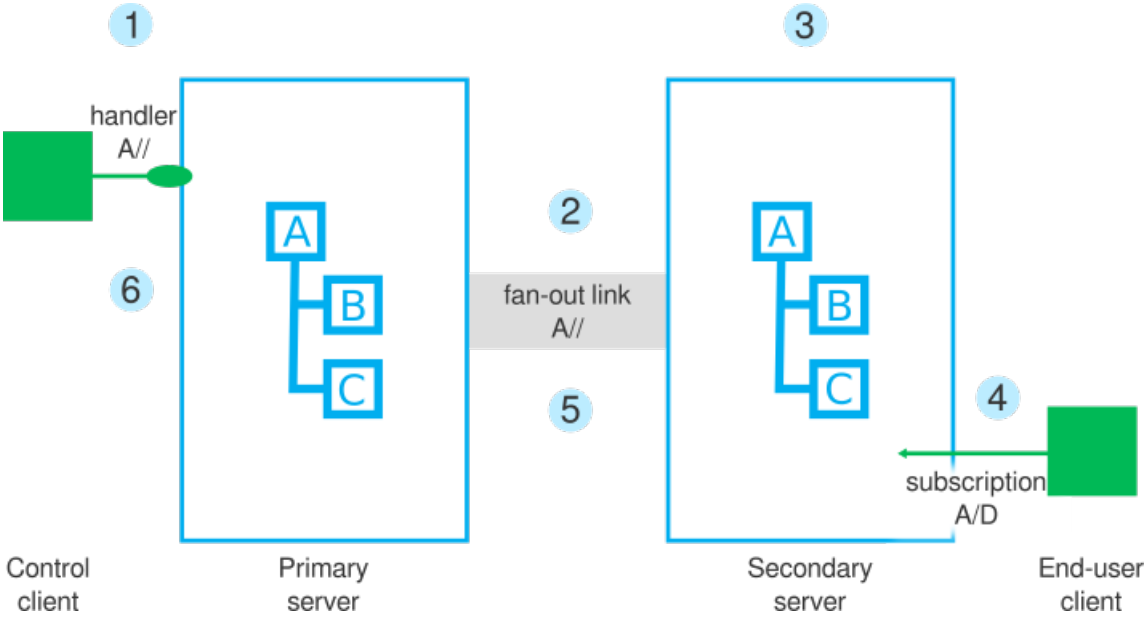


Figure 10: Missing topic notification propagation

1. A control client connects to the primary server and registers a missing topic notification handler against the A branch of the topic tree.
2. A secondary server connects to the primary server and replicates the A branch of the topic tree.
3. On the secondary server the replicated part of the topic tree comprises the following topics: A, A/B and A/C.
4. An end-user client attempts to subscribe to A/D, which does not exist.
5. The topic A/D is in part of the topic tree that is matched by a fan-out link selector, so the secondary server propagates the missing topic notification to the primary server.
6. The topic A/D does not exist on the primary server, so the primary server sends the missing topic notification to the handler registered by the control client.

Missing topic notification handlers at both the primary and secondary servers

A single subscription can cause a missing topic notification to be sent to a handler registered against the secondary server as well as a handler registered against a primary server.

The decision about whether to notify the handlers registered against a primary server is based on the intersection of the selector used by the subscription with the selector used to configure the fan-out link. It is possible for a missing topic notification to be sent to the primary server, but not to local handlers because the selector matches other (non-replicated) topics hosted by the secondary.

In particularly complex configurations, multiple primary servers might receive the notification or there can be multiple tiers of fan-out connections.

Considerations when using missing topic notifications with fan-out

The intersection of the selector used by the subscription request with a selector used for a fan-out link is calculated based only on the path-prefix of each selector. Complex selectors that use regular expressions can produce false positive results or false negative results. We recommend that you do not use regular expressions in the selectors used to configure fan-out links.

Ensure that the principal that the secondary server uses to make the fan-out connection to the primary server has the SELECT_TOPIC permission for the path prefix of the selector that triggered the missing topic notification.

A current session must exist between the secondary server and the primary server to forward notifications. If there is no session or the session fails while the missing topic notification is in-flight, the secondary server logs a warning message and discards the notification.

The robustness of the session between the servers can be improved by configuring reconnection. Fan-out connections can have a large number of messages in flight. It might be necessary to tune the reconnection time-out and increase queue depth and recovery buffer sizes.

Related concepts

Handling subscriptions to missing topics on page 254
A client can handle subscription requests for topics that do not exist and can act on those notifications, for example, by creating a topic on demand that matches the request.

Topic persistence

Consider if you want to enable topic persistence for fast recovery of topic state when Diffusion servers restart.

Topic persistence

Topic persistence enables a server to store the state of topics (and the topic tree) to the local file system as a special persistence log file. When the server restarts, topics are automatically restored to the state they were in when the server was stopped.

The persistence log only stores the most recent state of each topic.

Persistence avoids the need for the server to rebuild the topic tree from scratch when it starts. Persistence is useful to speed up development, as well as for backing up and restoring the state of topics at a particular point in time.

Persistence can be enabled or disabled for the whole server. As of Diffusion 6.1, when persistence is enabled for the server, it can still be disabled for individual topics using the PERSISTENT topic property.

Considerations

Consider the following factors when using topic persistence:

- Persistence is disabled by default.
- Each server has an append-only log file of topic events. By default, this is stored in a directory named `persistence` under the Diffusion server home directory.
- The log file is written until it reaches 200MiB in size. It is then switched out of service and automatically compacted to save storage space. Compaction removes redundant information (for example, if a topic has been removed, the history of values for that topic is removed during compaction, and only the last value is retained).
- Enabling persistence consumes a significant amount of storage space. You will need to allow 200MiB for the active log file, plus space for the compacted file. The size of the compacted file depends on the details of your application. The more topics you have, the bigger the file will be.
- As a rough indication of the size of the compacted file, it contains two records for each topic: a topic record and a value record. The size of each topic record will be approximately the length of the topic name plus 20 bytes. The size of each value record will be approximately the length of the value or delta plus 13 bytes. In practice, you should allow extra space and make sure to monitor the free space available on the server.
- You can configure the directory where the persistence files are stored. See Configuring topic persistence on page 422 for details.

- You should only back up or restore a persistence log while the Diffusion server is not running.
- Enabling persistence increases memory usage. The compaction service uses about as much memory as it takes to store the topics themselves.
- If servers close unexpectedly (for example, due to a crash), the persistence feature may not log the most recent topic updates, resulting in some data loss when the server restarts.
- Current information is always prioritised. During recovery from the file system, if a newer state of a topic is available (for example, from an application update or by replication from another server), that state will be used instead.
- If you do not want topics to be restored when you next start the server, simply delete all the log files.

Persistence and topic replication

In a cluster with topic replication enabled, topic values from other servers in the cluster take precedence over values recovered from persistence.

If only one server in the cluster receives a topic update, the updated value will not be recovered from the persistence file if another server in the cluster has a value for that topic.

When you restart a cluster, it is best to start with the server that was shut down last.

Alternatively, you can do a rolling restart and make sure that the cluster recovers between each server start.

Related tasks

Configuring topic persistence on page 422
Use the `Server.xml` configuration file to configure the topic persistence feature.

Topic delivery priority

You can set a delivery priority for each topic, which is used in case of an update backlog to a session.

Topic delivery priority

Diffusion is designed to deliver topic updates to each session as quickly as possible. At times, a backlog of updates to be delivered can develop. Situations where this can occur include:

- a topic is being updated at a high rate
- the Diffusion server or client are heavily loaded
- the client session is recovering from a temporary disconnection
- there is poor network connectivity between the Diffusion server and the client

In many applications, some topics are more time-sensitive than others. Topic delivery priority enables you to specify which topics should be prioritised if there is a backlog.

Priority is set with the PRIORITY topic property. The possible values are `high`, `default` and `low`.

The priority affects the order of topic updates sent to a subscribed client session. When there are multiple topic updates for topics with different priorities in a session's outbound queue, updates for `high` priority topics will be delivered first, followed by updates for `default` priority topics, followed by updates for `low` priority topics.

Topic subscription and unsubscription notifications are also delivered according to the topic delivery priority.

Considerations

Consider the following factors when using topic delivery priority:

- If you do not specify a delivery priority, the `default` priority will apply.
- Delivery priorities will have little effect in conditions of light load, when updates only remain in the outbound queue for a few milliseconds, so there is a lower chance of topic updates being reordered based on their priority.
- You can reduce the load on your clients by making sure that only the required data is being delivered. Consider using topic views to reduce the amount of topic updates each client receives. You can avoid sending updates for parts of the topic tree that are not relevant to the client, or apply a topic view throttling clause to limit the rate of updates to a client.
- Messages from the server to the client that are not topic updates, for example ping requests and responses, are queued with the `default` delivery priority.

Automatic topic removal

Automatic topic removal

You can use an automatic topic removal policy to remove topics when a set of conditions you specify is met.

A simple use case would be a policy that automatically removes a topic if it has not been updated for a day.

A policy can remove the topic it applies to, or a set of topics specified with a path selector. For example, a policy could say that if topic `a` has not been updated for a day, topics `b/1` and `b/2` get removed.

An automatic topic removal policy is set with the REMOVAL topic property. The property is specified with an expression including the conditions and the optional selector specifying topics to remove.

See Removing topics automatically on page 284 for details.

Considerations

Consider the following factors when using automatic topic removal:

- If persistence is not enabled for a topic, the topic will be removed when a server shuts down, regardless of the removal policy.
- If persistence is enabled for a topic, the topic's removal policy will persist. After a server is shut down and restarted, the removal policy will continue to be evaluated.
- Policies are evaluated every few seconds, so a time-based removal policy may not be applied at the exact second specified.
- Automatic topic removal is supported for replicated topics and for topics fanned out to secondary servers using the fan-out feature. The removal policy conditions will be evaluated as a whole across the cluster and at downstream fan-out servers.
- Removal policy conditions take into account reference topics created from the monitored topic by a topic view.
- If a reference topic created by a topic view is fanned out to a secondary server, sessions subscribed to the replica of the reference topic at the secondary server do not count towards the total subscriptions for the source topic with the removal policy.
- If your application requires a private topic for each user/principal, you can use Topic ownership on page 156 with an automatic topic removal policy to remove the topic when the user stops being active.

Clients

Consider how you use clients in your solution.

Clients are key to a Diffusion solution. Your solution must include clients as an endpoint to distribute data to. However, clients can also be used for control purposes.

When using clients in your solution, consider the following:

- What types of client you require
- What you use your clients for

Client types

Diffusion provides APIs for many languages and platforms. Some of these APIs have different levels of capability.

A client's type is a combination of the API it uses and the protocol it uses to connect to the Diffusion server.

Clients can also connect using the MQTT protocol instead of an API.

APIs

JavaScript

Use this API to develop browser or Node.js™ clients.

Apple

Use this API to develop clients in Objective-C for iOS, tvOS, or macOS.

Android

Use this API to develop mobile clients in Java.

Java

Use this API to develop Java clients.

.NET

Use this API to develop clients in C#.

C

Use this API to develop C clients for Linux, Windows, or macOS.

Python

Use this API to develop Python clients.

Protocols

The following protocols, and their secure versions, are available:

WebSocket

The WebSocket implementation provides a browser-based full duplex connection, built on top of WebSocket framing. This complies with the WebSocket standards and is usable with any load balancer or proxy with support for WebSocket.

HTTP Polling

The HTTP polling protocol uses HTTP requests with header m=1 to poll the server and HTTP requests with header m=2 to send messages to the server.

Table 17: Supported protocols by client

Client	WebSocket	HTTP Polling	
JavaScript API	✓	✓	
Apple API	✓		
Android API	✓	✓	
Java API	✓	✓	
.NET API	✓		
C API	✓		

MQTT clients can connect via WebSocket or TCP.

Using clients

Most clients connect to the Diffusion server only to subscribe to topics and receive message data on those topics. Some clients can also perform control actions such as creating and updating topics or handling events.

Subscribe to topics and receive data

The majority of clients that connect to the Diffusion server, do so to subscribe to topics and receive updates that are published to those topics. These are the clients used by your customers to interact with the data your organization provides.

Control Diffusion, other clients, or the data

You can also develop clients that control aspects of the Diffusion server, other clients, or the data distributed by Diffusion. These are the clients that are used by users or systems inside your organization.

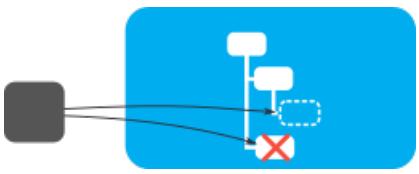
Using clients for control

Clients can perform control actions that affect the Diffusion server, other clients, or the data distributed by Diffusion.

> **Note:** Support for these control features can differ slightly between APIs. For more information, see the documentation for the feature.

When designing your Diffusion solution, decide whether you want to use clients to perform the following actions:

Create and delete topics



Clients can create any type of topic on the Diffusion server. These topics can be created explicitly or dynamically in response to a subscription request from another client.

These topics have the lifespan of the Diffusion server unless the client specifies that the topic be removed when the client session closes.

Clients can also delete topics from the Diffusion server.

You can also use publishers to create and delete topics.

For more information, see Managing topics on page 244.

Publish updates to topics

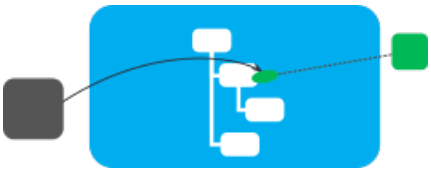
Clients can publish updates to topics that are pushed out to clients subscribed to that topic. These updates can be made exclusively, so that only one client can update a given topic, or non-exclusively, allowing any client to update a given topic.

> **Note:** Do not design your solution to require a large number of update sources (for example, do not give each topic its own exclusive topic updater).

You can also use publishers to publish updates to topics.

For more information, see Updating topics on page 291.

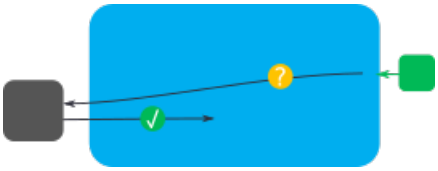
Subscribe other clients to topics



Clients can subscribe other client sessions to topics and also unsubscribe other client session from topics.

For more information, see Managing subscriptions on page 309.

Authenticate other clients



Clients can provide authentication decisions about whether to allow or deny other client sessions connecting to the Diffusion server. These clients can also assign roles to the connecting client sessions that define the permissions the connecting client has.

You can also use the system authentication handler or an authentication handler located on the Diffusion server to authenticate other clients.

For more information, see User-written authentication handlers on page 153.

Modify the security information stored on the Diffusion server

Clients can modify the information stored in the security store on the Diffusion server. The security store can be used to specify which permissions are assigned to roles and which roles are assigned to anonymous sessions, and named-principal sessions.

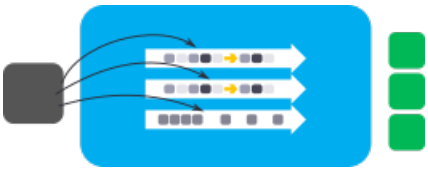
For more information, see Updating the security store on page 367.

Modify the authentication information stored on the Diffusion server

Clients can modify the information stored in the system authentication store on the Diffusion server. The system authentication store can be used to specify which principals a client session can use to connect and what roles are assigned to an authenticated client session.

For more information, see Updating the system authentication store on page 356.

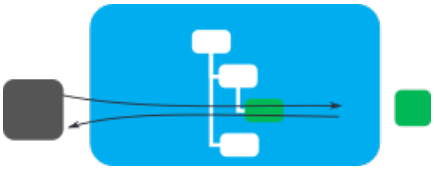
Manage the flow of data to clients



Updates are pushed to subscribing clients through client queues. Clients can receive notifications when client queues reach a certain threshold. These clients can manage the client queues by turning on throttling or conflation for the queue.

For more information, see Managing sessions on page 377.

To handle messages sent to message paths by clients and send messages to specific clients



Clients can send messages through message paths to specific clients. Clients can also register to handle messages that are sent to a message path. Messages sent using topic paths do not update the topic.

You can also use publishers to handle messages on message paths and send messages to clients.

For more information, see .

Client coordination

Clients can coordinate their access to shared resources using session locks.

Session locks

Session locks enable you to implement collaborative locking schemes to manage access to shared resources by multiple clients.

Each lock is identified by an arbitrary lock name and can only be owned by one session at a time.

The main reason to use a session lock is to ensure only one session can update a set of topics. You might also use a session lock to:

- ensure that at most one session responds to an event
- select a single session to perform a housekeeping task

See Session locks on page 293 for details of how to use session locks.

Considerations

Consider the following factors when using session locks:

- Session locks are a generic mechanism for you to implement a locking scheme. It is up to you to design a suitable scheme and ensure that each client or other application component follows it appropriately.
- Locks are only associated with sessions. There is no inherent way to associate a lock with a topic, except through your application's logic.
- If a lock is released, and multiple sessions are trying to acquire it, the server will arbitrarily assign the lock to one of the sessions.
- There is a potential for client-side race conditions to arise due to the distributed nature of session locks. Even if an application is coded correctly to protect a shared resource using session locks, there may be a period where two or more sessions concurrently 'believe' they have the lock.
- The server, or cluster of servers, is responsible for assigning the owner and has a single view of lock ownership. When locks are used with an update constraint to ensure a single session can update a set of topics, this view guarantees that only the current owner satisfies the constraint

User-written components

Consider deploying user-written components to define custom behavior as part of your solution.

Server-related components

All of these components must be created as Java classes and put on the classpath of the Diffusion server.

Authentication handlers

These components handle authentication of clients that connect to the Diffusion server or change the principal that they use to connect to the Diffusion server. If the client connection is allowed, the authentication handler assigns roles to the client session.

You can have zero, one, or many authentication handlers configured on your Diffusion server.

For more information, see Developing a local authentication handler on page 391 and Developing a composite authentication handler.

> **Note:** Local authentication handlers, on the Diffusion server, can be written only in Java. However, control authentication handlers that are part of a client whose API supports Authentication Control can be written in other languages.

Hooks

Startup and shutdown hooks are called by the Diffusion server. The startup hook is instantiated and called as the Diffusion server starts and before publishers are loaded. The shutdown hook is called as the Diffusion server stops.

For example, you can use a shutdown hook to persist some aspect of the state of the Diffusion server to disk.

HTTP service handlers

These components handle HTTP requests as part of an HTTP service in the Diffusion server's built-in web server. Provide a user-written HTTP service handler to enable the Diffusion web server to handle any kind of HTTP request.

Thread pool handlers

These handlers define custom behavior in the Diffusion server related to the inbound thread pool.

You can provide a rejection handler that customizes the behavior when a task cannot be run by the thread pool. By default, if a task cannot be run by the inbound thread pool — for example, if the thread pool is overloaded — the calling thread blocks until there is space on the queue.

You can provide a notification handler that receives notifications when events occur on the inbound thread pool.

Topic- and data-related components

All of these components must be created as Java classes and put on the classpath of the Diffusion server.

Message matchers

Message matchers are used to customize conflation behavior. These classes that define how the Diffusion server locates messages on a client's message queue that are to be conflated.

By default, messages for conflation are matched if they are on the same topic.

For more information, see Conflation on page 102.

Message mergers

Message mergers are used to customize conflation behavior. These classes that define how the Diffusion server conflates matching messages.

By default, the older of the matching messages is removed.

For more information, see Conflation on page 102.

Routing topic handlers

These components handle the behavior of a routing topic. When you create a routing topic, you provide a routing topic handler that, when a subscription to the routing topic is made, maps the routing topic to another topic on the Diffusion server on a client-by-client basis.

For more information, see Routing topics on page 86.

Third party components

Diffusion interacts with a number of third-party components. Consider how you use these components as part of your solution.

Load balancers

We recommend that you use load balancers in your Diffusion solution.

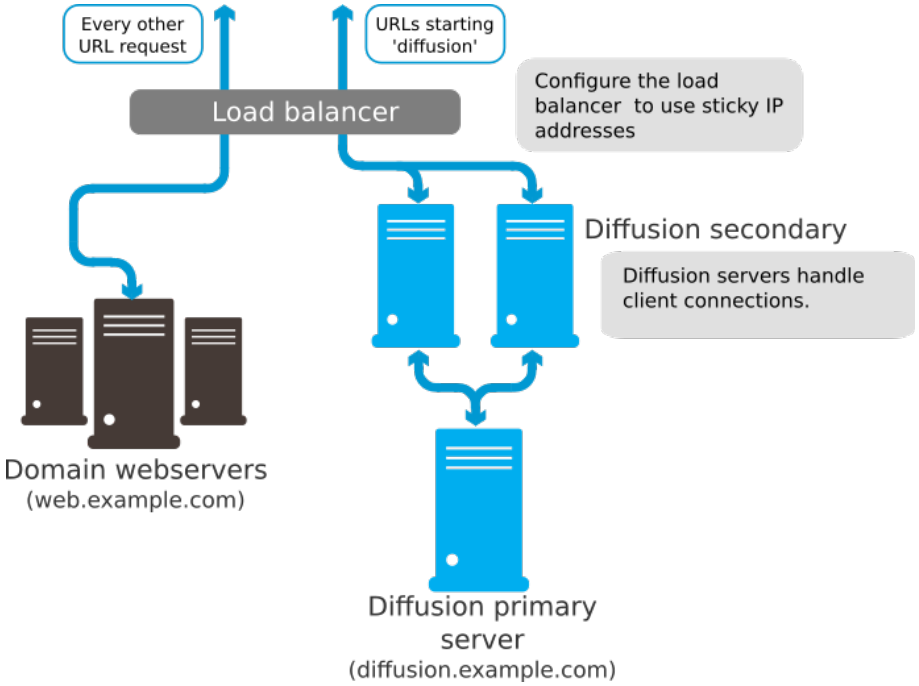
Why use load balancers?

Balancing client traffic across multiple Diffusion servers

Distribute incoming requests from clients fairly over the Diffusion servers in your solution and ensure that all traffic for a specific client is routed to the same Diffusion server.

Compositing URL spaces

If your Diffusion servers are located at a different URL to the Diffusion browser clients hosted by your web servers, you can use a load balancer to composite the URL spaces. This enables Diffusion solutions to interoperate with browser security.



SSL offloading

Diffusion clients can connect to your solution using TLS or SSL. The TLS/SSL can terminate at your load balancer or at your Diffusion server. Terminating the TLS at the load balancer reduces CPU cost on your Diffusion servers.

Considerations when using load balancers

Do not use connection pooling for connections between the load balancer and the Diffusion server. If multiple client connections are multiplexed through a single server-side connection, this can cause client connections to be prematurely closed.

In Diffusion, a client is associated with a single TCP/HTTP connection for the lifetime of that connection. If a Diffusion server closes a client, the connection is also closed. Diffusion makes no distinction between a single client connection and a multiplexed connection, so when a client sharing a multiplexed connection closes, the connection between the load balancer and Diffusion is closed, and subsequently all of the client-side connections multiplexed through that server-side connection are closed.

Multiple users masquerading behind a proxy or access point can have the same IP address, and all requests from clients with that IP address are routed to the same Diffusion instance. Load balancing still occurs, but some hosts might be unfairly loaded.

Web servers

Consider how to use web servers as part of your Diffusion solution.

If you are using Diffusion in conjunction with a web client or web application, this web client or application must be hosted on a web server.

While the Diffusion server includes a web server, this internal web server is intended for the following uses:

- Hosting the Diffusion landing page, demos, and management console
- Providing an endpoint for the HTTP-based transports used by Diffusion clients
- Optionally, hosting a static page you can use the check the status of the Diffusion server

For more information, see Diffusion web server on page 618.

Do not use the Diffusion web server as the host for your production website. Instead use a third-party web server.

There are two ways you can use Diffusion with a third-party web server:

- As separate, complementary components in your solution.
- With the Diffusion server deployed inside a web application server.

Use a separate web server with the Diffusion server

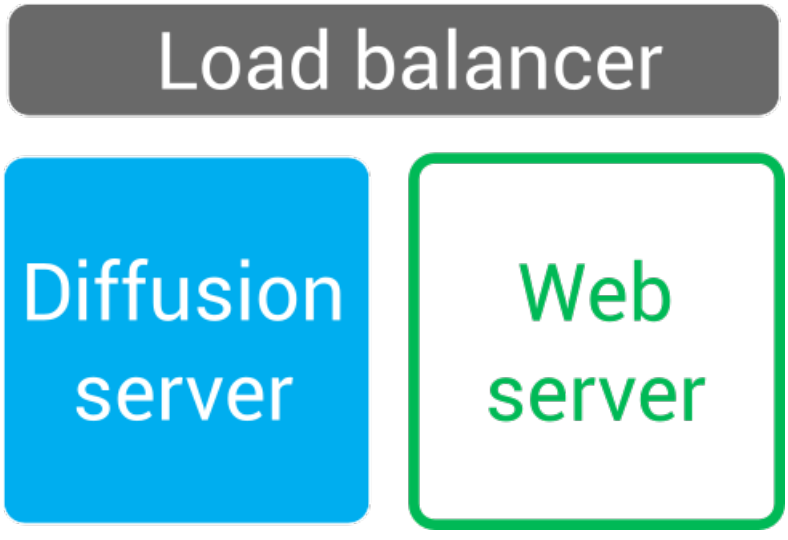


Figure 11: Using a web server with Diffusion

Why use a separate web server with Diffusion?

You can use a third-party web server to host your Diffusion browser clients.

A third-party web server provides the following advantages over the lightweight internal Diffusion web server:

- Greater ability to scale
- More comprehensive security
- Server-side code and dynamic web pages

If your organization already uses a third-party web server, Diffusion augments this component instead of replacing it.

Using a separate web server with the Diffusion server provides the following advantages over deploying the Diffusion server inside a web application server:

- The load balancer set up is simpler
- You can scale the number of Diffusion servers and the number of web servers in your solution independently and more flexibly
- The web server and the Diffusion server do not share a JVM process, which can cause performance advantages
- The web server and the Diffusion server are independent components, which makes them unlikely to be affected by any problems that occur in the other component

For more information, see Hosting Diffusion web clients in a third-party web server on page 620.

Considerations when using a separate web server with the Diffusion server

If your web server hosts a client that makes requests to a Diffusion server in a different URL space, you can use a load balancer to composite the URL spaces and interoperate with browser security or you can set up cross-domain policy files that allow requests to the different URL space.

When the Diffusion server is separate from the web server, the web server has no access to the Diffusion Server API.

Deploy the Diffusion server inside a web application server

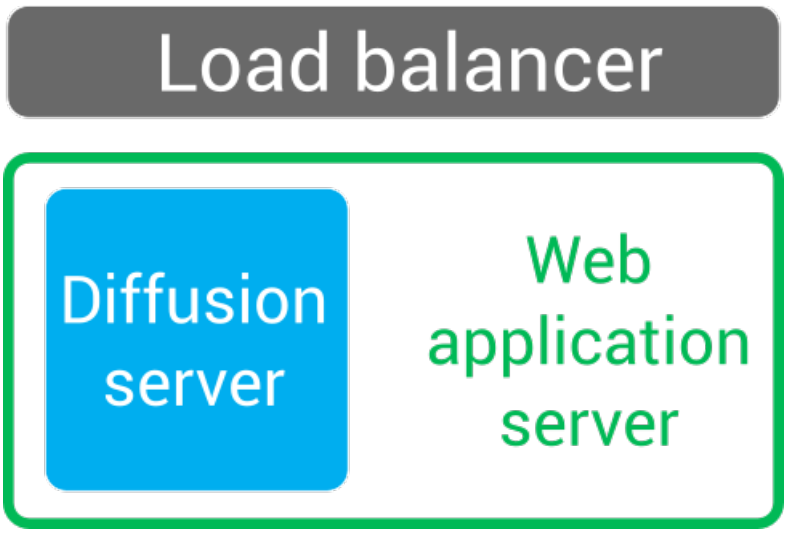


Figure 12: Deploying Diffusion inside a web application server

Why deploy Diffusion inside a web application server?

You can also host your Diffusion server inside a third-party web application server that has the capability to host Java servlets.

This provides the advantage of only setting up a single server and having a single application to manage when hosting your web application.

The web application server has access to the Diffusion Server API of the Diffusion server is hosts. This enables your web application to use server-side logic to include Diffusion information in your web pages.

For more information, see Running the Diffusion server inside of a third-party web application server on page 621.

Considerations when deploying the Diffusion server inside a web server

When running inside a web application server, the Diffusion server still requires its own internal web server to communicate with clients over HTTP-based transports.

Your web application and your Diffusion server, while hosted by the same server, can have different port numbers. This can result in cross-origin security concerns for some browsers. You can use a load balancer to composite the ports or you can set up cross-domain policy files that allow requests to the different ports.

The load balancer configuration can be more complex when deploying the Diffusion server inside a web application server. If you have multiple web application server and Diffusion server pairs, configure your load balancer to ensure that requests from a client always go to a pair and not to the web application server from one pair and the Diffusion server from another pair.

When running the Diffusion server inside of a web application server, the Diffusion server and the web application server share a JVM process. This can lead to large GC pauses. Ensure that you test this configuration and tune the JVM

Related concepts

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Configuring Diffusion web server security on page 473
When configuring your Diffusion web server, consider the security of your solution.

Running the Diffusion server inside of a third-party web application server on page 621
Diffusion can run as a Java servlet inside any Java application server.

Hosting Diffusion web clients in a third-party web server on page 620
Host Diffusion web clients on a third-party web server to enable your customers to access them.

Related reference

WebServer.xml on page 474

This file specifies the schema for the web server properties.

JMS

Consider whether to incorporate JMS providers into your solution.

If a third-party JMS provider is part of your solution, you can map JMS queues and topics to Diffusion topics by using the Diffusion JMS adapter.

We support integration with JMS providers that conform to version 1.1 or later of the JMS specification.

The following JMS products have been tested and are certified by Push Technology for use with the JMS adapter:

- Apache ActiveMQ v5.11
- IBM MQ v8

Why use a third-party JMS provider

If you are already using a JMS provider to move data in your internal system, you can integrate it with Diffusion to distribute that data to clients and users outside of your organization.

Diffusion JMS adapter

The JMS adapter for Diffusion, enables Diffusion clients to transparently send data to and receive data from destinations (topics and queues) on a JMS server. It is highly configurable and can support the following scenarios:

Pub-sub

Messages on a JMS destination can be published to a Diffusion topic. For more information, see Publishing using the JMS adapter.

Messaging

Messages can be sent between JMS destinations and Diffusion clients.

- A message on a JMS destination can be relayed to a Diffusion client through a topic path.
- A Diffusion client can send a message on a message path and the message is relayed to a JMS destination.

For more information, see Sending messages using the JMS adapter.

Request-response

The JMS provider can integrate with services that interact using an asynchronous request-response pattern. Diffusion exposes these JMS services through its messaging capabilities. For more information, see Using JMS request-response services with the JMS adapter.

Data that flows between JMS and Diffusion must be transformed. JMS messages contain headers, properties, and a payload. Diffusion messages contain just content. For more information about how data is transformed between JMS and Diffusion, see Transforming JMS messages into Diffusion messages or updates.

Running the JMS adapter in the Diffusion server or as a standalone application

The JMS adapter is provided in the following forms:

Within the Diffusion server

The JMS adapter can be configured to run as part of the Diffusion server process. A JMS adapter running within the Diffusion server cannot become disconnected from the Diffusion server.

As a standalone client

The JMS adapter is a Java application that can be run on any system and acts as a client to the Diffusion server. Topics created by the JMS adapter running as a standalone client are not deleted from the Diffusion server if the JMS adapter becomes disconnected. You can use this capability to design a highly available solution.

For more information, see JMS adapter on page 639.

Considerations when using the JMS adapter

Topics defined and created by the JMS adapter when it runs within the Diffusion server are removed when the JMS adapter is stopped.

Topics defined and created by the JMS adapter when it runs as a standalone client are not deleted from the Diffusion server when the JMS adapter client session is closed.

The JMS adapter supports interaction with Diffusion topics that are either stateful (single value) or stateless topics.

Only textual content and JMS TextMessages are supported. Binary content is not currently supported.

You cannot currently publish data to a Diffusion topic and have it sent to a JMS destination.

Data must be transformed between JMS messages and Diffusion content.

If multiple Diffusion servers subscribe to the same JMS queue in a request-response scenario, there is the risk of one server consuming messages intended for another server. Use JMS selectors to ensure that the JMS adapter only receives those messages intended for it.

The creation of temporary queues and topics by the JMS adapter is not currently supported.

Durable subscriptions are not supported.

JMS transactions are not supported.

The only acknowledgment mode that is supported is AUTO_ACKNOWLEDGE.

Session properties are not currently supported. The exception is the `$Principal` property.

Related concepts

Transforming JMS messages into Diffusion messages or updates

Publishing using the JMS adapter

Configuring the JMS adapter on page 640

Use the `configuration.json` configuration file to configure the JMS adapter to send and receive messages with destinations on a JMS server.

Example solutions

This section includes some example solutions that you can refer to when designing your own solution.

Example: Simple solution

This solution uses a firewall to restrict incoming traffic and a load balancer to balance the traffic between multiple Diffusion servers.

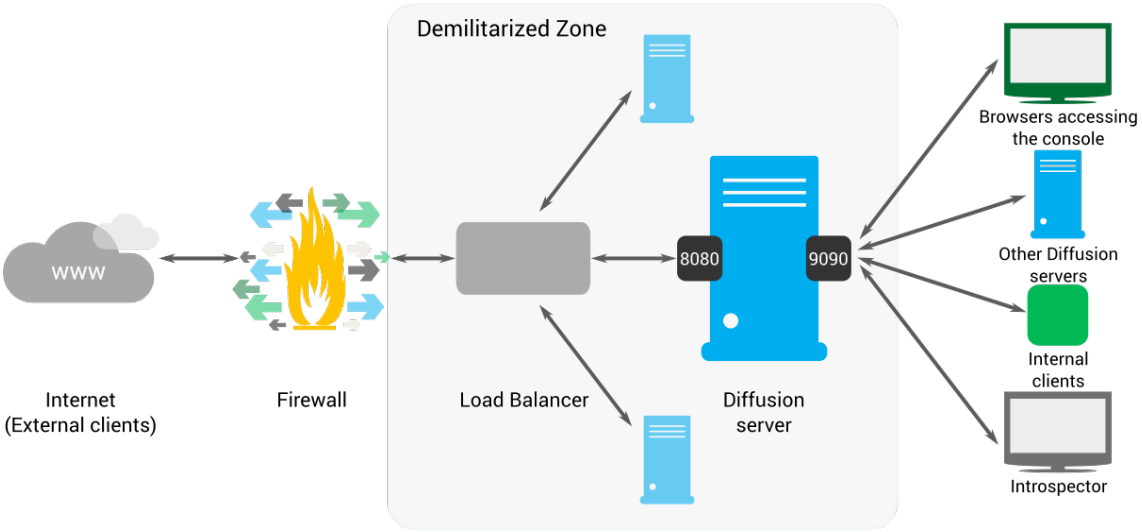


Figure 13: A simple solution

- Client applications can connect to Diffusion from the internet through a firewall.
- The firewall protects the DMZ from unwanted traffic. It allows connections on port 80 and redirects these connections to port 8080.
- The load balancer balances the Diffusion connections between all the Diffusion servers in the DMZ. You can also use the load balancer to filter the URL space and to perform SSL offloading.
- The Diffusion servers receive connections from external clients on port 8080. This port is protected by an authentication handler that performs strict authentication on the incoming connections. Authentication handlers can be local to the server or part of a control client.
- The Diffusion servers receive connections from internal clients on another port, for example 9090. The authentication controls on this port are less strict because these connections come from within your network. Internal connections can come from any of the following components:
  - Browsers accessing the Diffusion console
  - Internal clients, such as control clients.

Example: A solution using clients

Clients with different uses connect to the Diffusion server in this example solution.

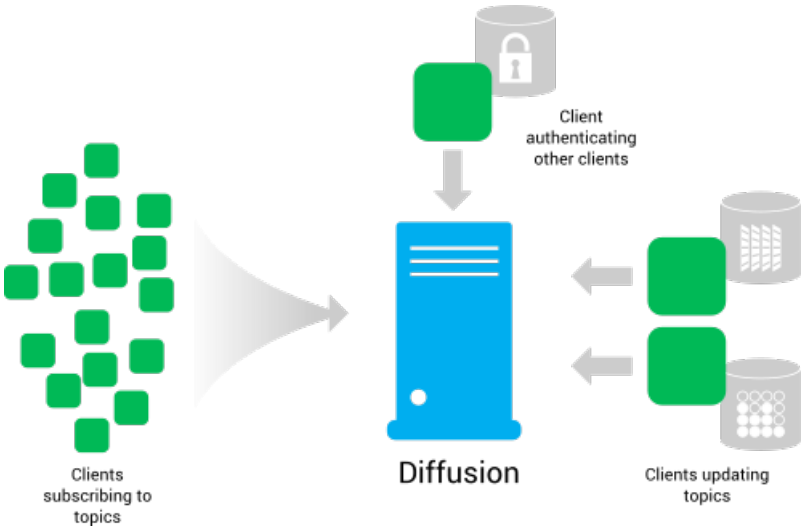


Figure 14: Clients for different purposes

This example solution uses three kinds of client, each for a different purpose:

Clients subscribing to topics

These clients are used by your customers to receive the data you distribute. You can use any of the provided APIs to create these, depending on how your customers want to access your data. For example,

- Use the Apple API to create an iPhone app.
- Use the JavaScript API to create a browser client.

These clients subscribe to the topics that are of interest to your customer, receive updates published on these topics, and display the information to your customers.

Clients creating and updating topics

These clients are used by your organization to distribute your data. You must use an API that provides control features to create these clients. For example, the JavaScript API or the .NET API.

These clients create the topics required to support your data structure and to publish data from your data sources to topics on the Diffusion server.

Clients authenticating other clients

These clients are used by your organization to authenticate connections from other clients. You must use an API that provides control features to create these clients. For example, the Java API.

These clients are called by the Diffusion server to provide an authentication decision when another client connects to the Diffusion server anonymously or with a principal. In addition to deciding whether the other client is allowed to connect, the authenticating client can assign roles to the client session.

The authenticating client can use information stored elsewhere in your system, for example in an LDAP server, to make the authentication decision and assign roles.

Example: Scalable and resilient solution

This solution uses replication to share information between primary servers and make them highly available. The solution also uses fan-out to spread the data from the primary servers behind the firewall to secondary servers in the DMZ.

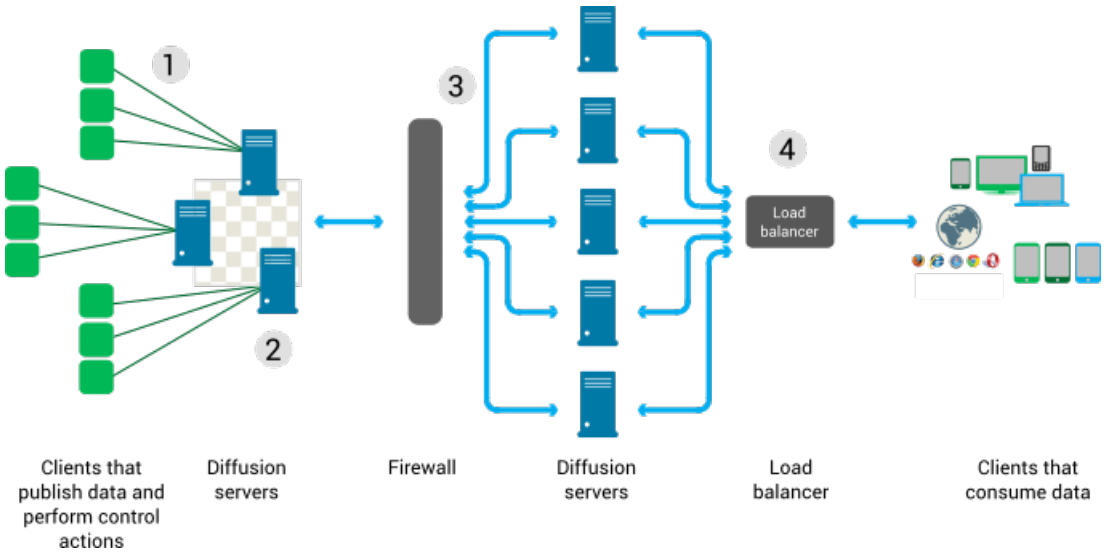


Figure 15: Architecture using replication and fan-out

1. Three clients register handlers with each of the Diffusion servers behind the firewall. These clients can be located on the same system as the server or on remote systems. Each Diffusion server load balances requests between clients that have registered to handle requests of that type. If one of the clients becomes unavailable, the requests can be directed to another client. You can connect more client sessions to deal with higher volumes of requests.
2. The Diffusion servers inside the firewall replicate information into a datagrid. If a Diffusion server that was handling a client session or topic becomes unavailable, the responsibility for that client session or topic can be passed to another Diffusion server that has access to all the information for that session or topic through the datagrid.
3. The Diffusion servers outside of the firewall, in the DMZ, are configured to use automated fan-out to connect to the Diffusion servers inside the firewall. Specified topics on the primary server are fanned out to the secondary servers.
4. You can use a load balancer to spread requests from subscribing clients across many secondary Diffusion servers. If a server becomes unavailable, clients can be directed to another server.

Security

Diffusion secures your data by requiring client sessions to authenticate and using role-based authorization to define the actions that a client can perform.

Concepts

Principal

The principal is a user or system user that has an identity that can be authenticated.

When a principal is authenticated is becomes associated with a session. The default principal that is associated with a session is ANONYMOUS.

Session

A session is a set of communications between the Diffusion server and a client.

Permission

A permission represents the right to perform an action on the Diffusion server or on data.

Role

A role is a named set of permissions and other roles. Principals and sessions can both be assigned roles.

Role hierarchy

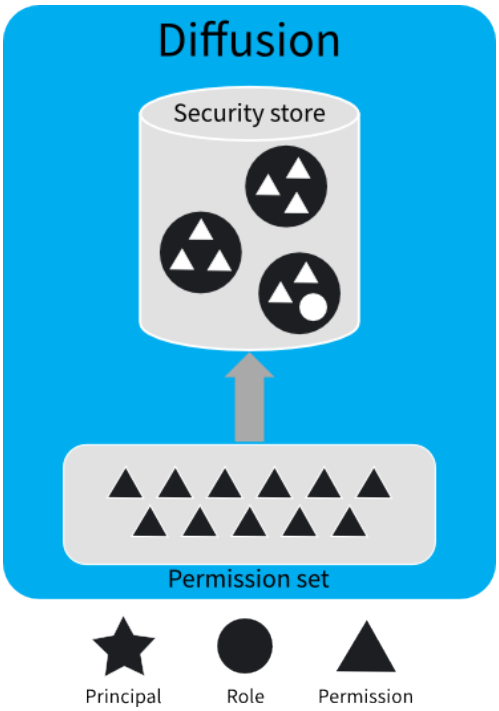
Roles are hierarchical. A role can include other roles and, by doing so, have the permissions assigned to the included roles. A role cannot include itself, either directly or indirectly – through a number of included roles.

Role-based authorization

Diffusion restricts the ability to perform actions to authorized principals. Roles are used to map permissions to principals.

Associating permissions with roles

The association between roles and permissions is defined in the security store.



A fixed set of permissions is defined by the Diffusion server. These permissions are used to control access to actions and data on the Diffusion server.

Roles are used to associate permissions to principals. Permissions are assigned to roles, and roles are assigned to principals.

A role can be assigned zero, one, or many permissions. The same permission can be assigned to multiple roles. Roles can also include other roles to form a role hierarchy, and so inherit their permissions. The permissions assigned to a role and the role hierarchy are defined in the security store.

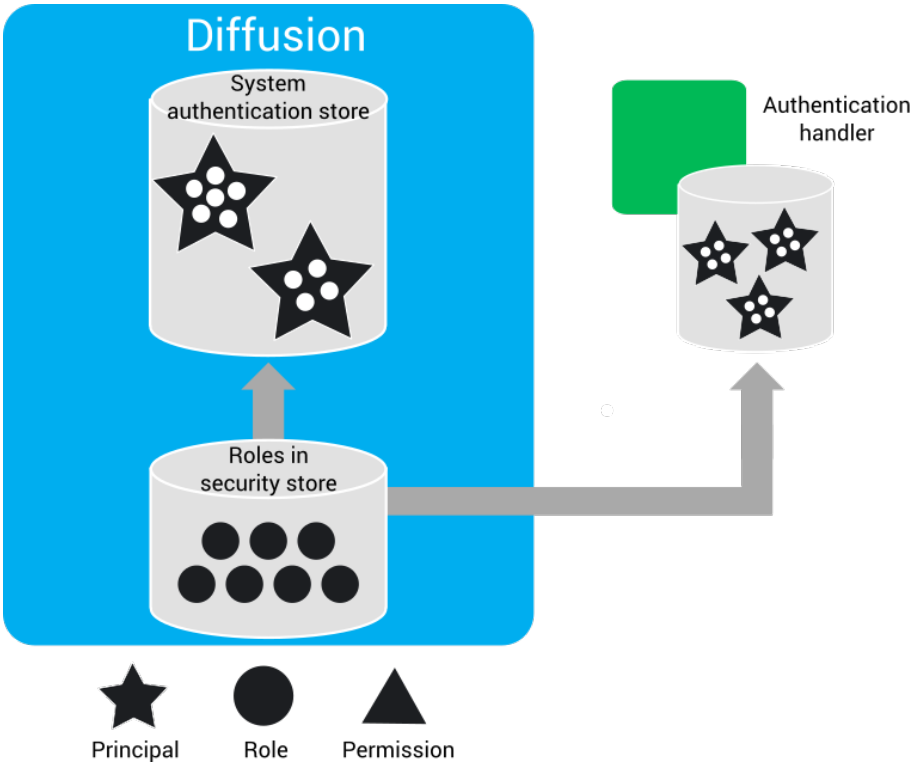
You can update the security store from a client using the SecurityControl feature.

You can update the security store by editing the store file, *`installation_directory`*`/persistence/Security.store`, and restarting the Diffusion server. Note that if you have never started the server, the file is not found in the `persistence` directory; an initial file is in *`installation_directory`*`/etc`, and this is copied into the `persistence` directory when you first start the server.

You should stop the server before editing the system authentication store directly. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

Associating roles with principals

The association between roles and principals is defined in the system authentication store or by user-written authentication handlers.



The association between principals and roles is defined in the following ways:

- In a user-defined store that your user-written authentication handlers refer to. For example, an LDAP server.
- A user-written authentication handler can also hard code the relationship between principals and roles, if that is appropriate.
- In the system authentication store of the Diffusion server

  The system authentication store is designed to hold information about Diffusion administration users and system clients. It can manage hundreds or perhaps thousands of principals, but does not provide the administration tools necessary to support millions of principals. We recommend that you delegate such "internet scale" use cases to a third-party identity provider using a custom authentication handler. For example, by using the OAuth or OpenID protocol.

You can update the system authentication store in the following ways:

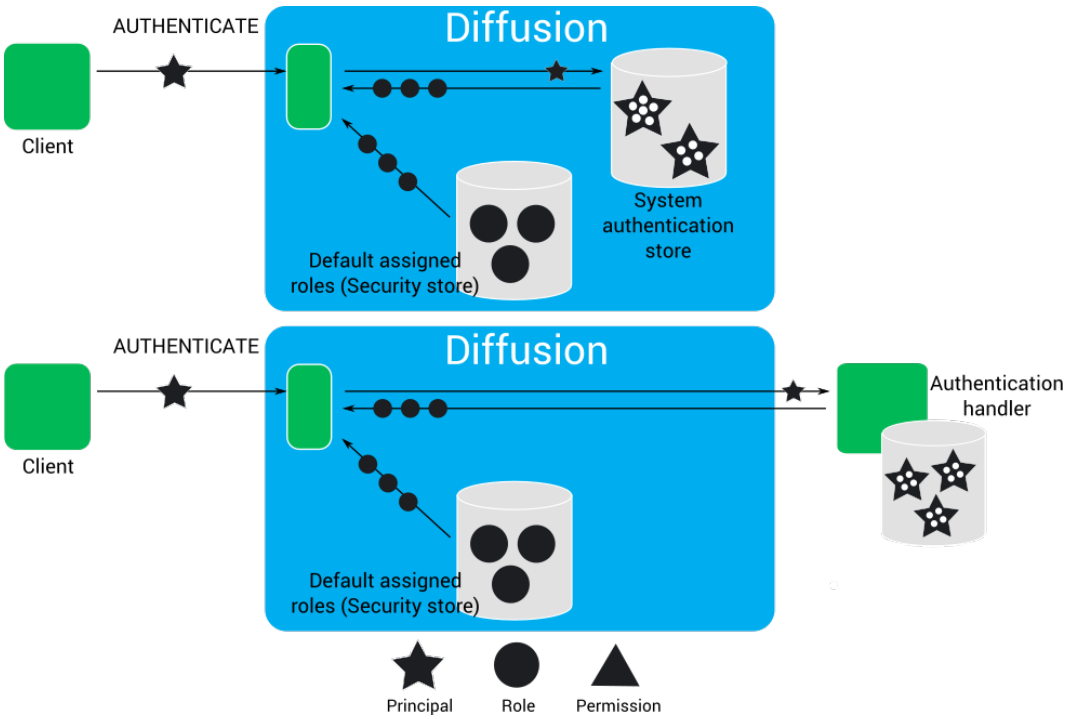
- From a client using the SystemAuthenticationControl feature.

- By editing the store file, by default found at *installation_directory*/persistence/SystemAuthentication.store, and restarting the Diffusion server. Note that if you have never started the server, the file will not be found in the persistence directory; an example file will be found in *installation_directory*/etc, and this is copied into the persistence directory when you first start the server.

It is recommended that you update the store from a client instead of editing the file directly. Do not edit the store file if you are using clustered Diffusion servers.

Assigning roles to client sessions

Roles are assigned to a new client session after client authentication.



The roles assigned to a client session determine the actions that client session can perform.

A client session is assigned roles in the following ways:

- If the client session connects to the Diffusion server anonymously, the client session is assigned the default assigned roles for the ANONYMOUS principal.

  Anonymous authentication can be enabled or disabled in the system authentication store. If enabled, roles can also be specified.
- When a client session authenticates with a principal, the client session can be assigned the following roles:
  - The default assigned roles for a named principal.
  - The set of roles assigned to a principal by the authentication handler that accepts the client session's authentication request. This authentication handler can be one of the following types:
    - The system authentication handler, in which case the roles that are assigned are those associated with that principal in the system authentication store.
    - A user-written authentication handler, in which case the roles that are assigned are those defined by the handler or a user-defined store.
- A client session with the correct privileges can change the security roles assigned to another session. This requires modify_session and view_session permissions.

For example: A client session authenticates with the Diffusion server using the principal Armstrong. The first authentication handler that is called is a user-written authentication handler. This authentication handler abstains from the authentication decision, so does not assign roles to the client session. The next authentication handler that is called is the system authentication handler. The system authentication handler does not abstain from the authentication decision. It uses the information in the system authentication store to decide to allow the authentication request. In the system authentication store, the principal Armstrong is associated with the roles ALPHA, BETA, and EPSILON. These roles are assigned to the client session.

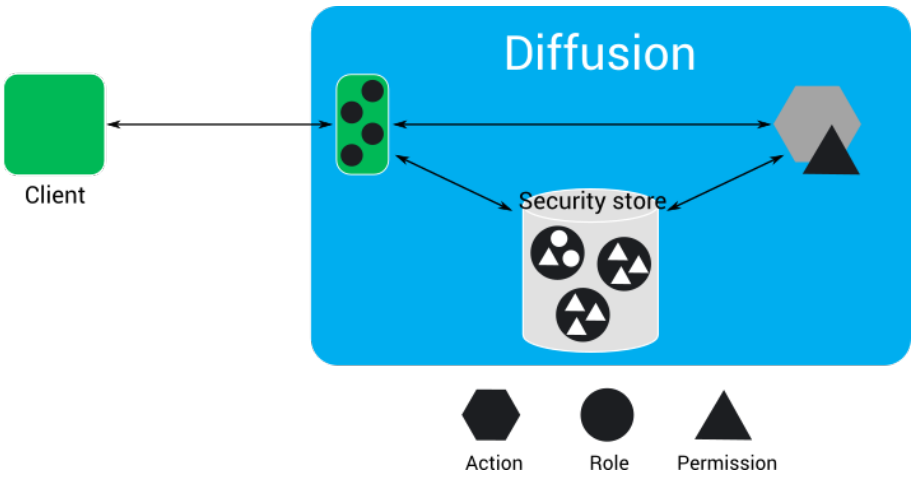
After the authentication request has been allowed, no further authentication handlers are called to make a decision or assign roles. However, the Diffusion server also assigns the default assigned roles for a named principal to the client session. The default assigned roles defined in the security store are GAMMA and RHO.

After authenticating with the principal Armstrong, the client session has the following roles assigned to it:

- ALPHA
- BETA
- EPSILON
- GAMMA
- RHO

Authorizing actions

When a client requests to perform an action or access data that requires a permission, the Diffusion server checks whether the client session is assigned a role that includes the required permission.



The client requests to perform an action. If the action requires that the client session has a permission, the Diffusion server checks what roles the client session is assigned and checks in the security store whether any of these roles have the required permission.

For example: A client requests to subscribe to the topic A/B/C. To subscribe to a topic, a client session must have the select_topic permission for that topic. The client session has the ALPHA and BETA roles. In the security store, the ALPHA role does not include the select_topic permission, but the BETA role does include the select_topic permission for the A/B/C topic. Because the client session is assigned the BETA role, it has the required permission and can subscribe to the topic.

Related concepts

Authentication on page 149

You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

DEPRECATED: Authorization handlers on page 157
An authorization handler can control authorization and permissions for clients and users.

Related reference

Topic ownership on page 156
Topic ownership allows you to grant read, modify and update permissions for a topic to a specific principal. This is useful to provide a 'private' topic for a given user.

Configuring security on page 446
The `Security.store` file contains the rules that determine the permissions for each role. The `SystemAuthentication.store` stores the principals recognised by the system authentication handler.

Permissions

The actions a client session can take in Diffusion are controlled by a set of permissions. These permissions are assigned to roles.

Permissions can have one of the following scopes:

Path

Permissions at path scope apply to actions on a topic path, request-response message path or session lock name.

Path-scoped permissions are defined against paths. The permissions that apply are the set of permissions defined at the most specific path.

Global

Permissions at global scope apply to actions on the Diffusion server.

Path permissions

The path-scoped permissions are listed in the following table:

Table 18: List of path-scoped permissions

Name	Description
acquire_lock	Acquire a session lock. The names of the locks that can be acquired are restricted to the paths of the permission scope.
select_topic	Use a topic selector that selects the topic path. A session must have this permission for the path prefix of any topic selector used to subscribe or fetch.
read_topic	Grant read access to the topics. If a session does not have this permission for a topic, that topic does not match subscriptions and is excluded from fetch requests. Also, the topic's details cannot be retrieved. Changes to the security store which alter read_topic permission assignments are applied dynamically. This means that if you change the permissions granted by a role, the new

Name	Description
	configuration is immediately applied to all sessions. Each session's topic selections are re-evaluated with the new permissions, and subscriptions are added or removed accordingly.
query_obsolete_time_series_events	Evaluate a query on a time series topic that can potentially return a non-current view of all or part of a time series. Such queries include value range queries that specify an edit range, and all types of edit range query. Evaluating a query also requires read_topic.
edit_time_series_events	Submit edit events to a time series topic. Updating a time series topic also requires update_topic.
edit_own_time_series_events	Submit edit events to a time series topic where the event author is the same as the principal of the calling session. Updating a time series topic also requires update_topic.
update_topic	Update topics at or below a topic branch.
modify_topic	Create or remove topics at or below a topic branch.
send_to_message_handler	Send a message to the Diffusion server through a message path.
send_to_session	Send a message to a client session through a message path.

Configuring path permissions

Path permissions are stored as rules in the system security store. You should create and update them via a client using the API, or using the Diffusion management console. See Configuring security on page 446 and the API documentation for your SDK for more details.

Understanding path permissions

Path permissions enable you to control access to resources that have a path-based hierarchy. The most common example is the topic tree, but request-response messaging paths and session lock names also use path permissions.

A session can carry out an action on a resource if any of its roles grants the appropriate permission for the resource path.

Each role assigns parts of the path hierarchy to permissions. The permissions a role grants for a path are determined by the assignment with the longest prefix of the path. For a given role, only one assignment applies to a path.

For example, suppose a session has two path permission rules for topics:

- read_topic permission for the topic path `telemetry/gps`
- read_topic and update_topic permission for the path `telemetry/gps/ships`

Consider the topic `telemetry/gps/submarines/nautilus`. Only the first rule matches the path, so the session would have read_topic permission for the topic.

Consider the topic `telemetry/gps/ships/titanic`. Both rules match, but the second rule is more specific, so the session has both read_topic and update_topic permission.

A session has a permission if any of its assigned roles has that permission (or includes a role that has that permission).

> **Note:** This way of combining permissions from is a change from how it worked before version Diffusion 6.5. If you upgrade from 6.4 to 6.5, your existing configuration will be rewritten automatically to provide the same behavior as before. See Upgrading from version 6.4 to version 6.5 on page 666 for details.

You can also define default path permissions. These are used if there are no matching path permissions and the path is not isolated (see below).

Isolating branches of the path hierarchy

Normally, if a role has no permission assignment for a path, the permissions are inherited from the permission assignment for the longest prefix of the path.

For example, if a role has no permission assigned for `telemetry/gps/balloons`, but it does have one for `telemetry/gps`, `telemetry/gps/balloons` will normally inherit the permissions.

Sometimes you may not want this inheritance to apply to a particular branch. For example, suppose your topic tree has this branch, with sensitive data that should only be accessed by sessions with a special role:

`telemetry/gps/ships/glomar-explorer`

If you grant another role read_topic permission to `telemetry/gps/`, sessions with that role will be able to read the sensitive branch.

One workaround would be to avoid creating any path permissions above `telemetry/gps/ships/`, but that means creating many more specific path permissions and updating them every time a new ship is added.

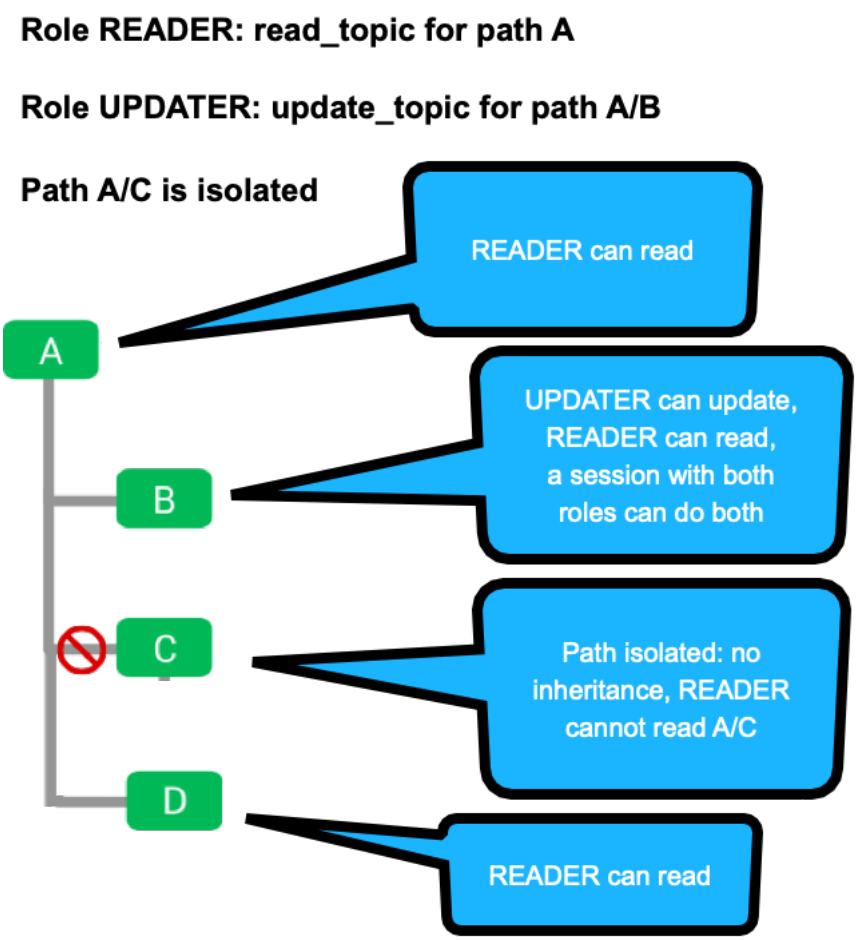
Isolating paths is a way to disable the usual inheritance rules and ignore default path permissions.

You can use the client API or the management console to create a rule that isolates a specified path.

In the above example, suppose you isolate `telemetry/gps/ships/glomar-explorer`. Now a role that has read_topic permission to `telemetry/gps/` will not grant the ability to read the `glomar-explorer` topic or any of its children, such as `telemetry/gps/ships/glomar-explorer/location`.

In addition, a default path permission granting read_topic does not apply to the isolated path.

Path scope example



This example demonstrates how path permissions and isolated paths combine.

- The READER role has `read_topic` permission for path A, so a session with just this role can read the content of topic A. It can also read A/B and A/D, because the first part of the path matches and there are no conflicting permission rules.
- The UPDATER role has `update_topic` permission for path A/B only.
- A session with both READER and UPDATER roles combines the permissions above, granting `read_update` and `update_topic` permissions for A/B, as well as read access to A and A/D.

  Note that if you had used a single role and granted `read_update` to A and `update_topic` to A/B, the session would not have both permissions to A/B, because only the more specific rule would be applied.
- The path A/C is isolated, so normal inheritance rules do not apply. The READER role cannot read A/C (or any child topics like A/C/E).

The security store rules to configure this scenario would be:

```
set "READER" path "A" permissions [READ_TOPIC]
set "UPDATER path "A/B" permissions [UPDATE_TOPIC]
isolate path "A/C"
```

See DSL syntax: security store on page 368 for details.

Understanding the select_topic and read_topic permissions

The default configuration grants the select_topic and read_topic permissions to CLIENT sessions including anonymous sessions. You can alter this configuration to protect sensitive topics.

A session that does not have the select_topic permission for a particular path cannot subscribe directly to topics at that path. However, the session can be independently subscribed to that topic by a control session that has modify_session permission in addition to the select_topic permission for that path. The subscribed session requires the read_topic permission for that topic for the subscription to the topic to occur. The control session cannot subscribe a session to a topic if that session does not have the read_topic permission for the topic. When this occurs, the topic is filtered out of the subscription.

Managing all subscriptions from a separate control session

You can prevent client sessions from subscribing themselves to topics and control all subscriptions from a separate control client session that uses SubscriptionControl feature to subscribe clients to topics.

To restrict subscription capability to control sessions, configure the following permissions:

Control session:

- Grant the modify_session permission
- Grant the select_topic permission

  This can either be granted for the default path scope or more selectively to restrict the topic selectors the control session can use.

Other sessions:

- Grant read_topic to the appropriate topics.
- Deny the select_topic permission by default.

  Do not assign the session a role that has the select_topic permission for the default path scope. This prevents the session from subscribing to all topics using a wildcard selector.
- Optionally, grant the select_topic permission to specific branches of the topic tree to which the session can subscribe freely.

Global permissions

The global permissions are listed in the following table:

Table 19: List of global permissions

Name	Description
view_session	List or listen to client sessions.
modify_session	Alter a client session. This covers a range of actions including the following: • subscribe a session to a topic • throttle a session • enable conflation for a session • close a session
register_handler	Register any handler with the Diffusion server.
authenticate	Register an authentication handler.

Name	Description
	The register_handler permission is also required to perform this action.
view_server	Read administrative information about the Diffusion server. For example, through JMX.
control_server	Server control functions: • create or modify remote server definitions • create or modify metric collectors • shut down the Diffusion server (via JMX) • start and stop and remove connectors (via JMX) • start and stop HTTP services (via JMX) • clear the HTTP file service cache (via JMX)
view_security	View the security policy.
modify_security	Change the security policy.
read_topic_views	View the topic views.
modify_topic_views	Change the topic views. To add a new topic view the session also needs the select_topic permission for the prefix of the source selector of the topic view being added.

Related reference

Pre-defined roles on page 147
Diffusion has a pre-defined set of roles with associated permissions.

Pre-defined users on page 155
Diffusion has a pre-defined set of users with associated password and roles.

Pre-defined roles

Diffusion has a pre-defined set of roles with associated permissions.

Clients can edit this set of roles. For more information, see Updating the security store on page 367.

	CLIENT	TOPIC_ CONTROL	CLIENT_ CONTROL	AUTHENTICATION _HANDLER	OPERATOR	ADMINISTRATOR and JMX_ ADMINISTRATOR
acquire lock Default scope		✓	✓			✓
acquire lock "Diffusion" topic						

	CLIENT	TOPIC_ CONTROL	CLIENT_ CONTROL	AUTHENTICATION _HANDLER	OPERATOR	ADMINISTRATOR and JMX_ ADMINISTRATOR
select topic Default scope	✓	✓	✓	✓	✓	✓
select topic "Diffusion" topic					✓	✓
read topic Default scope	✓	✓	✓	✓	✓	✓
read topic "Diffusion" topic					✓	✓
modify topic Default scope		✓				✓
modify topic "Diffusion" topic						✓
update topic Default scope		✓				✓
update topic "Diffusion" topic						✓
query_obsolete_time_series_events						
edit_time_series_events						
edit_own_time_series_events						
send to message handler Default scope	✓	✓	✓	✓	✓	✓
send to message handler					✓	✓

	CLIENT	TOPIC_ CONTROL	CLIENT_ CONTROL	AUTHENTICATION _HANDLER	OPERATOR	ADMINISTRATOR and JMX_ ADMINISTRATOR
"Diffusion" topic						
send to session Default scope		✓				✓
view session			✓		✓	✓
modify session			✓			✓
register handler			✓	✓		✓
authenticate				✓		
view security						✓
modify security						✓
view server					✓	✓
control server						✓
read topic views		✓				✓
modify topic views		✓				✓

Related reference

Permissions on page 142
The actions a client session can take in Diffusion are controlled by a set of permissions. These permissions are assigned to roles.

Pre-defined users on page 155
Diffusion has a pre-defined set of users with associated password and roles.

Authentication

You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

The handlers you can implement and register are the following:

- Any number of local authentication handlers

- Any number of control authentication handlers

The server calls the authentication handlers (local and control) in the order that they are defined in the `Server.xml` file.

If no handlers are defined, the server allows the client operation by default.

Authentication process

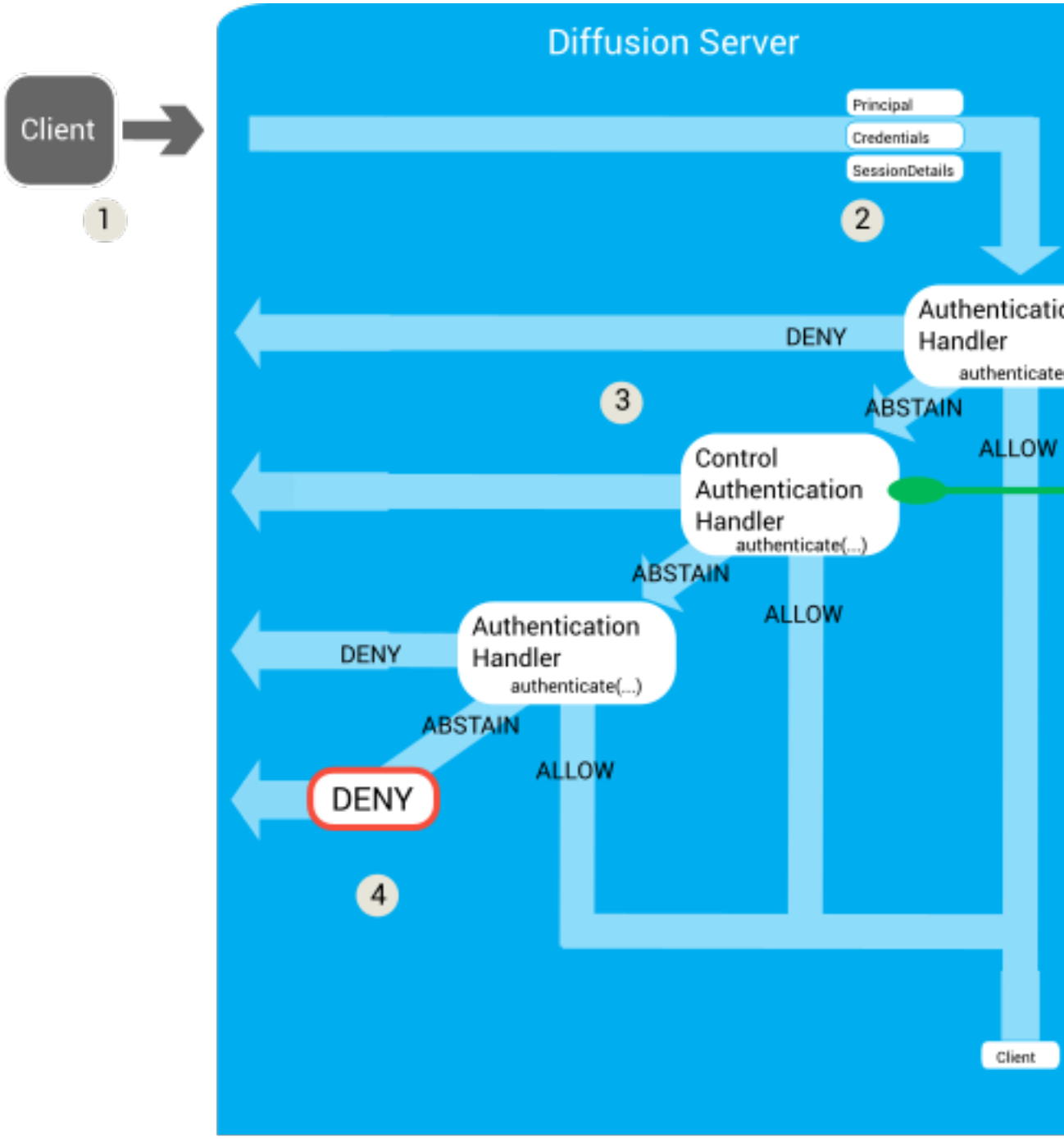


Figure 16: Authentication process for clients

1. A client tries to perform an operation that requires authentication. For more information, see Client operations that require authentication on page 152.
2. The server calls the authentication handlers one after another in the order that they are listed in the `Server.xml` file. It passes the following parameters to each authentication handler's `authenticate()` method:

   Principal

A string that contains the name of the principal or identity that is connecting to the server or performing the action. This can have a value of `Session.ANONYMOUS`.

Credentials

The `Credentials` object contains an array of bytes that holds a piece of information that authenticates the principal. This can be empty or contain a password, a cryptographic key, an image, or any other piece of information. The authentication handler is responsible for interpreting the bytes.

SessionProperties

This contains information about the client. The available properties depend on what information the server holds about the client session.

This information can be used in the authentication decision. For example, an authentication handler can allow connection only from clients that connect from a specific country.

When it registers with the server, a control authentication handler can specify what properties it requires, so only these properties are sent by the server (if available). This reduces the amount of data sent across the control client connection.

The authentication handler is passed two property maps.

- The `sessionProperties` map contains the fixed properties for the session, and if re-authenticating, can also contain the user-defined properties of the current session.
- The `proposedProperties` map contains any user-defined properties proposed by the client when opening the session. This map is empty when re-authenticating.

Client-proposed session properties can provide additional information about the client that can be used later to select or filter sessions. They can also provide credential information for use in the authentication decision.

Callback

A callback that the authentication handler can use to respond to the authentication request by using the callback's `allow()`, `deny()`, or `abstain()` method.

If the authentication handler is a local authentication handler, the authentication logic is done on the server. If the authentication handler is a control authentication handler, the parameters are passed to a control client and the control client handles the authentication logic and returns a response.

3. Each authentication handler can return a response of `ALLOW`, `DENY`, or `ABSTAIN`.
   - If the authentication handler returns `DENY`, the client operation is rejected.
   - If the authentication handler returns `ALLOW`, the decision is passed to the authorization handlers. The authentication handler can also provide a list of roles to assign to the client session.
   - If the authentication handler returns `ABSTAIN`, the decision is passed to the next authentication handler listed in the `Server.xml` configuration file.
4. If all authentication handlers respond with an `ABSTAIN` decision, the response defaults to `DENY`.

Client operations that require authentication

The following client operations require authentication with the server:

Table 20: Client operations that require authentication

Client operation	Behavior if operation is allowed	Behavior if operation is denied
Connect to server	The client connection to the server is accepted.	The client connection to the server is rejected and is dropped.
Change the principal associated with a client session	The principal is changed.	The principal is not changed, but the client session is not dropped.

Related concepts

Role-based authorization on page 138
Diffusion restricts the ability to perform actions to authorized principals. Roles are used to map permissions to principals.

DEPRECATED: Authorization handlers on page 157
An authorization handler can control authorization and permissions for clients and users.

Related reference

Topic ownership on page 156
Topic ownership allows you to grant read, modify and update permissions for a topic to a specific principal. This is useful to provide a 'private' topic for a given user.

User-written authentication handlers

You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

The authentication handlers can be implemented either remotely, in a client, or locally, on the server. The authentication handlers can be individual authentication handlers, that perform a single authentication check, or composite authentication handlers, that delegate to one or more individual authentication handlers.

Local authentication handlers

A local authentication handler is an implementation of the `Authenticator` interface. Local authentication handlers can be implemented only in Java. The class file that contains a local authentication handler must be located on the classpath of the Diffusion server.

Control authentication handlers

A control authentication handler can be implemented in any language where the Diffusion API includes the AuthenticationControl feature. A control authentication handler can be registered by any client that has the authenticate and register_handler permissions.

For more information, see Authenticating new sessions on page 342.

The following table matrix shows the types of authentication handler.

Table 21: Types of authentication handler

	Individual
Local	Implement the `Authenticator` interface. For more information, see Developing a local authentication handler on page 391.

	Individual
Control	Implement the `ControlAuthenticator` interface. For more information, see Developing a control authentication handler on page 352.

Related concepts

Configuring authentication handlers on page 420
Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

Related tasks

Developing a local authentication handler on page 391
Implement the `Authenticator` interface to create a local authentication handler.

Developing a composite authentication handler

Developing a control authentication handler on page 352
Implement the `ControlAuthenticator` interface to create a control authentication handler.

Developing a composite control authentication handler

Developing a local authentication handler on page 391
Implement the `Authenticator` interface to create a local authentication handler.

Developing a composite authentication handler

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

System authentication handler

Diffusion provides an authentication handler that uses principal, credential, and roles information stored in the Diffusion server to make its authentication decision.

System authentication store

The principal, credentials, and role information located in the system authentication store is used by the system authentication handler to authenticate users.

The system authentication store is designed to hold information about Diffusion administration users and system clients. It can manage hundreds or perhaps thousands of principals, but does not provide the administration tools necessary to support millions of principals. We recommend that you delegate such "internet scale" use cases to a third-party identity provider using a custom authentication handler. For example, by using the OAuth or OpenID protocol.

By default the following information is set in the system authentication store file, `SystemAuthentication.store`. This file is located in the `persistence` directory. If the server has never been started, the example file in `etc` is copied into `persistence` on first starting the server.

```
allow anonymous connections [ "CLIENT" ]

add principal "client" "password" [ "CLIENT" ]
add principal "control" "password" [ "CLIENT_CONTROL" "TOPIC_CONTROL"
 "AUTHENTICATION_HANDLER" ]
add principal "admin" "password" [ "ADMINISTRATOR" ]
add principal "operator" "password" [ "OPERATOR" ]
```

You can edit the usernames and passwords in this file by hand and restart the Diffusion server to reload the file. However, any password you enter in plaintext is hashed by the Diffusion server when it starts and the plaintext value in this file is replaced with the hashed value.

Do not edit the file manually if you are using clustered servers.

The default hash scheme used is PBKDF-SHA256-1000. You can specify a different hash scheme in the `Server.xml` configuration file.

Behavior of the system authentication handler

The system authentication handler behaves in the following way:

- If anonymous connections are allowed in the system authentication store and a client session connects anonymously, the system authentication handler returns an ALLOW decision and the list of roles an anonymous client session is assigned.
- If anonymous connections are not allowed in the system authentication store and a client session connects anonymously, the system authentication handler returns a DENY decision.
- If a client session connects with a principal listed in the system authentication store and the correct credentials, the system authentication handler returns an ALLOW decision and the list of roles that client session is assigned.
- If a client session connects with a principal listed in the system authentication store and incorrect credentials, the system authentication handler returns a DENY decision.
- If a client session connects with a principal that is not listed in the system authentication store, the system authentication handler returns an ABSTAIN decision.

Related concepts

Configuring authentication handlers on page 420
Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

Updating the system authentication store on page 356
A client can use the SystemAuthenticationControl feature to update the system authentication store. The information in the system authentication store is used by the system authentication handler to authenticate users and assign roles to them.

Pre-defined users

Diffusion has a pre-defined set of users with associated password and roles.

You can use the SystemAuthenticationControl feature to edit this set of users.

> **Note:** This set of users and passwords are well known and not secure. Change the passwords or remove the users before putting Diffusion into production.

The users defined in the system authentication store are only authenticated if the system authentication handler is configured. For more information, see Configuring authentication handlers on page 420.

User	Password	Associated roles
client	password	CLIENT
control	password	CLIENT_CONTROL, TOPIC_CONTROL, AUTHENTICATION_HANDLER
admin	password	ADMINISTRATOR
operator	password	OPERATOR

User	Password	Associated roles
Anonymous connections		CLIENT

Related reference

Pre-defined roles on page 147
Diffusion has a pre-defined set of roles with associated permissions.

Permissions on page 142
The actions a client session can take in Diffusion are controlled by a set of permissions. These permissions are assigned to roles.

Topic ownership

Topic ownership allows you to grant read, modify and update permissions for a topic to a specific principal. This is useful to provide a 'private' topic for a given user.

Topic ownership enables you to give a specific security principal special access to a topic when the topic is created. Ownership grants `READ_TOPIC`, `MODIFY_TOPIC` and `UPDATE_TOPIC` permissions for the topic to sessions authenticated with that principal.

Use the OWNER topic property to set ownership.

The format of the property value is:

```
$Principal is "name"
```

where *name* is the name of the principal.

Topic ownership is useful in cases where your application requires a user to have privileged access to their own special topic. For example, it might be a private topic that only the user should see, or each user might have their own topic used to broadcast status information to other users.

All other sessions will only have the permissions granted by the security store.

Automatic topic removal can be used to remove a principal's private topic if that principal no longer has an active session for a period of time.

Topic ownership and topic views

Topic views respect the OWNER property when evaluating source topics. As a result, a source topic is selected for evaluation by a topic view if the principal of the topic view is the same as the owner of the source topic.

The OWNER property (if present) is copied from a source topic to all reference topics that are derived from it. Granting ownership of a source topic also grants ownership of all derived reference topics.

Both the topic view principal and the OWNER property (if set) are treated as owners of a reference topic. This means that the principal of the view which created a reference topic has full access to it, as well as any OWNER principal specified.

Related concepts

Role-based authorization on page 138
Diffusion restricts the ability to perform actions to authorized principals. Roles are used to map permissions to principals.

Authentication on page 149

You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

DEPRECATED: Authorization handlers on page 157
An authorization handler can control authorization and permissions for clients and users.

DEPRECATED: Authorization handlers

An authorization handler can control authorization and permissions for clients and users.

Role-based authorization

> **Attention:**
>
> The new role-based security model has superseded authorization handlers. Role-based security enables you to more simply manage permissions and users. We recommend you use role-based authorization instead of authorization handlers. For more information, see Role-based authorization on page 138.
>
> As of Diffusion 6.2, authorisation handlers can no long validate credentials.

An authorization handler is a user-written Java class that must implement the `AuthorisationHandler` interface in the .

Such a handler can be used to restrict access of clients according to any criteria that is appropriate. One capability within Diffusion is for a client to be able to specify Credentials when they connect that can be checked by the authorization handler.

The handler can either be specified in `etc/Server.xml` in which case it is loaded when the server starts or can be set programmatically within a publisher using the `Publishers.setAuthorisationHandler` method.

There can only be one handler and it is system wide across all publishers, although you can have authorization at the publisher level.

If an authorization handler is not specified, credentials sent by a client are assumed to be valid. A publisher has access to the credentials to perform finer-grained authorization, if required.

The authorization handler interface has the following methods:

Table 22: Authorization handler methods

`canSubscribe(Client, Topic)`	This method is called when a client subscribes to a topic. If topic information is sent with the connection, this method is called after the `canConnect` method. **Note:** This is called for every topic being subscribed to, even if subscribed as a result of a topic selector being specified. However (by default), if a topic is rejected by this method, it is not called again for any children (or descendants) of the topic.

`canSubscribe(Client, TopicSelector)`	This method is called when a client attempts to subscribe to a topic selector pattern (as opposed to a simple topic name). If topic information is sent with the connection, this method is called after the `canConnect` method.
`canFetch(Client, Topic)`	This method is called when a client sends a fetch request to obtain the current state of a topic. **Note:** This is called for every topic being fetched, even if fetched as a result of a topic selector being specified. However (by default), if a topic is rejected by this method, it is not be called again for any children (or descendants) of the topic.
`canWrite(Client, Topic)`	This method is called when a client sends a message on a given topic, if false is returned the message is ignored, and the publisher will not be notified of the message. When implementing this method, be aware that performance can be impacted if many clients send messages or if a few clients send large messages.

Authorization handler

Authorization at the publisher level can also be achieved. This is required if there are many publishers running within the same Diffusion Server and they have different security settings. The following code example works if the publishers all implement `AuthorisationHandler`

```
public boolean canSubscribe(Client client, Topic topic) {

    AuthorisationHandler handler =

 (AuthorisationHandler)Publishers.getPublisherForTopic(topic);

    // Call the publisher in question
    return handler.canSubscribe(client, topic);
}
```

Permissions

The permissions process governs whether a client is able to send messages to a publisher, or in other words, is the topic read only. This is handled by the `canWrite` method. Again a good pattern might be to look at the credentials attachment object to see if this is permissible.

```
public boolean canWrite(Client client, Topic topic) {
 User user = (User) client.getClientCredentials().attachment();
 return user.canWriteMessages(topic);
}
```

Related concepts

Role-based authorization on page 138
Diffusion restricts the ability to perform actions to authorized principals. Roles are used to map permissions to principals.

Authentication on page 149

You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

Related reference

Topic ownership on page 156
Topic ownership allows you to grant read, modify and update permissions for a topic to a specific principal. This is useful to provide a 'private' topic for a given user.

Part IV

Developer Guide

This guide describes how to develop clients, publishers, and server-side components that interact with the Diffusion server.

Diffusion clients

The Diffusion client API is a consistent and modular API that provides an asynchronous and session-oriented approach to developing your clients.

The Diffusion client API is available for the following platforms:

- JavaScript
- .NET
- Java
- Android
- C
- Python
- Apple

In this section:

- Best practice for developing clients
- Feature support in the Diffusion API
- Client basics
- Connecting to the Diffusion server
- Receiving data from topics
- Managing topics
- Updating topics
- Using time series topics
- Managing subscriptions
- Using request-response messaging
- Authenticating new sessions
- Updating the system authentication store
- Updating the security store

- Managing sessions
- Configuring conflation
- Logging from the client
- Developing other components
- Using Maven to build Java Diffusion applications
- Testing
- Creating a remote server definition

Best practice for developing clients

Follow these best practises to develop resilient and well performing clients.

Use an asynchronous programming model

All calls in the Diffusion API are asynchronous. Ensure that you code your client using asynchronous models to gain the advantages this provides.

Asynchronous calls remove the possibility of your client becoming blocked on a call. The Diffusion API also provides context-specific callbacks, enabling you to pass contextual information with a callback, and a wide range of event notifications.

Write good callbacks

The Diffusion API invokes callbacks using a thread from Diffusion thread pool. Callbacks for a particular session are called in order, one at a time. Consider the following when writing callbacks:

- Do not sleep or call blocking operations in a callback. If you do so, other pending callbacks for the session are delayed. If you must call a blocking operation, schedule it in a separate application thread.
- You can use the full Diffusion API to make other requests to the Diffusion server. If you want to make many requests based on a single callback notification, be aware that Diffusion client flow control is managed differently in callback threads. Less throttling is applied and it is easier to overflow the Diffusion server by issuing many thousands of requests. If you have a lot of requests to make, it is better to schedule the work in an application thread.

Use a modular design

The Diffusion API provides interfaces on a feature-by-feature basis. There is a clear delineation between features. At runtime, the client starts only those services that it uses.

You can take advantage of the modular design of the Diffusion API by designing multiple smaller and more modular control clients. Smaller modules are easier to design, maintain and keep running. Develop separate clients for different control responsibilities. For example, have a client or set of clients responsible for authentication and a different client or set of clients responsible for creating topics.

Also consider separating the responsibility for different parts of the topic tree between clients. For example, have a client or set of clients responsible for updating the Tennis branch of the topic tree and a different client or set of clients responsible for updating the Rugby branch of the topic tree.

Make your client resilient and defensive

If the Diffusion server restarts, all topic information — tree structure and topic state — is removed, all subscription information is removed, and all clients are disconnected. Security and authentication information is persisted.

If your client disconnects and cannot reconnect to the same session, all of its subscriptions and any handlers it has registered are lost.

Ensure that you program your clients to handle and respond to these possibilities.

Feature support in the Diffusion API

Review this information when designing your clients to determine which APIs provide the functionality you require.

Features are sets of capabilities provided by the Diffusion API. Some features are not supported or not fully supported in some APIs.

The Diffusion libraries also provide capabilities that are not exposed through their APIs. Some of these capabilities can be configured.

> **Note:** The Python client currently implements limited capabilities. See the Python API documentation for details.

Table 23: Capabilities provided by the Diffusion client libraries

Capability	JavaScript	Apple	Android	Java	.NET	C
Connecting to the Diffusion server on page 186						
Connecting on page 188	✓	✓	✓	✓	✓	✓
Connecting with multiple transports on page 189	✓	✗	✓	✓	✗	✗
Asynchronous connections on page 190	✓	✓	✓	✓	✓	✓
Synchronous connections on page 191	✗	✗	✓	✓	✓	✓
Connecting on page 188	✗	✓	✓	✓	✓	✓
Connecting on page 188	✓	✗	✓	✓	✗	✓
Connect securely on page 194	✓	✓	✓	✓	✓	✓
Configure the SSL context or behavior on page 194	✓	✓	✓	✓	✓	✗
Connecting through an HTTP proxy on page 199	✗	✓	✓	✓	✓	✗
Connecting through a load balancer on page 202	✓	✓	✓	✓	✓	✓
Connecting through a load balancer on page 202	✓	✓	✓	✓	✗	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Reconnect to the Diffusion server on page 203						
Reconnect to the Diffusion server on page 203	✓	✓	✓	✓	✓	✓
Configuring reconnection on the client on page 203	✓	✓	✓	✓	✓	✓
Specifying a reconnection strategy on page 206	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Reliable reconnection on page 204	✓	✓	✓	✓	✓	✓
Configuring the recovery buffer on the client on page 205	✗	✓	✓	✓	✓	✓
Monitoring the connection activity on page 206	✓	✓	✓	✓	✓	✓
Using TCP state on page 206	✓	✓	✓	✓	✓	✓
Pinging the Diffusion server on page 218	✓	✓	✓	✓	✓	✓
Change the security principal and credentials associated with your client session on page 219	✓	✓	✓	✓	✓	✓
Receiving data from topics						
Subscribe to a topic or set of topics	✓	✓	✓	✓	✓	✓
Using streams for subscription on page 227	✓	✓	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Using streams for subscription on page 227	✓	✓	✓	✓	✓	✓
Fetching the current value of a topic on page 233	✓	✓	✓	✓	✓	✓
Managing topics						
Managing topics on page 244	✓	✓	✓	✓	✓	✓
Slave topics (DEPRECATED) on page 87	✓	✓	✓	✓	✓	✓
Using time series topics on page 297	✓	✓	✓	✓	✓	✓
Create a topic and set initial value	✓	✓	✓	✓	✓	✓
Adding topics with topic specifications on page 245	✓	✓	✓	✓	✓	✓
Adding topics with topic specifications on page 245	✓	✓	✓	✓	✓	✓
Create a topic with metadata	✓	✓	✓	✓	✓	✓
Listen for topic events (including topic has subscribers and topic has zero subscribers)	✗	✓	✓	✓	✓	✗
Receiving topic notifications on page 240	✓	✓	✓	✓	✓	✓
Delete a topic	✓	✓	✓	✓	✓	✓
Delete a branch of the topic tree	✓	✓	✓	✓	✓	✓
Automatic topic removal on page 122	✓	✓	✓	✓	✓	✓
Updating topics						
Updating topics on page 291	✓	✓	✓	✓	✓	✓
Perform exclusive updates	✓	✓	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Perform non-exclusive updates	✓	✓	✓	✓	✓	✗
Managing subscriptions						
Managing subscriptions on page 309	✓	✓	✓	✓	✓	✓
Managing subscriptions on page 309	✓	✓	✓	✓	✓	✗
Managing subscriptions on page 309	✗	✗	✓	✓	✓	✗
Handling subscriptions to missing topics on page 254	✓	✓	✓	✓	✓	✓
Using request-response messaging on page 323						
Sending request messages to a message path on page 325	✓	✓	✓	✓	✓	✓
Sending request messages to a session on page 331	✓	✓	✓	✓	✓	✓
Sending request messages to a session filter on page 338	✓	✓	✓	✓	✓	✓
Sending request messages to a session on page 331	✓	✓	✓	✓	✓	✓
Sending request messages to a message path on page 325	✓	✓	✓	✓	✓	✓
Managing security						
Authenticating new sessions on page 342	✓	✗	✓	✓	✓	✓
Updating the system authentication store on page 356	✓	✗	✓	✓	✓	✓

Capability	JavaScript	Apple	Android	Java	.NET	C
Updating the security store on page 367	✓	✗	✓	✓	✓	✓
Updating the security store on page 367	✓	✗	✓	✓	✓	✓
Topic ownership on page 156	✓	✓	✓	✓	✓	✓
Managing other clients						
Receiving notifications of client session events and their session properties on page 378	✓	✗	✓	✓	✓	✓
Getting properties of specific client sessions on page 382	✓	✗	✓	✓	✓	✓
Update the properties of specific client sessions on page 383	✓	✗	✓	✓	✓	✓
Receiving notifications of client queue events on page 383	✗	✗	✓	✓	✓	✗
Conflate and throttle clients	✗	✓	✓	✓	✓	✗
Closing client sessions on page 377	✗	✓	✓	✓	✓	✗
Other capabilities						
Flow control on page 384	✗	✗	✓	✓	✓	✓

Client basics

Get started developing Diffusion clients by downloading one of our SDKs, discovering its capabilities, and starting to stream realtime data through the Diffusion server.

JavaScript

The JavaScript API is provided in the file `diffusion.js` and can be accessed through the web or through NPM.

Use with Node.js:

Install with npm:

```
npm install diffusion
```

Include in your Node.js application:

```
var diffusion = require('diffusion');
```

You can also download the JavaScript file as a tarball package that can be installed locally by using NPM:

```
https://download.pushtechnology.com/clients/6.7.7/js/diffusion-
js-6.7.7.tgz
```

Include JavaScript in a web page:

```
<script src="https://download.pushtechnology.com/clients/6.7.7/js/
diffusion-6.7.7.js"/>
```

This hosted version of the Diffusion JavaScript library is served with GZIP compression enabled. GZIP compression reduces the library to 20% of its uncompressed size and ensuring fast page loads.

Get the minified JavaScript:

Download the latest JavaScript file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/js/
diffusion-6.7.7.js
```

The JavaScript file is also located in your Diffusion server installation:

```
diffusion_directory/clients/js
```

The Diffusion JavaScript client library is a full featured library and as such is provided as a large download file. However, when served with GZIP compression, the size of the served file is significantly smaller than the size of the downloaded file. Ensure that you enable GZIP compression on the web server that hosts the JavaScript client library.

The minified version of the JavaScript client library is approximately 70% of the size of the unminified version.

Get the unminified JavaScript:

Download the latest unminified JavaScript client library from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/js/diffusion-unminified-6.7.7.js
```

The unminified JavaScript file is also located in your Diffusion server installation:

```
diffusion_directory/clients/js
```

The minified version of the Diffusion JavaScript client library is created with Browserify. The minified version might not be compatible with certain JavaScript frameworks. This unminified version is provided to enable you to include Diffusion in projects using any framework.

The unminified form of JavaScript client library also gives you the option to perform minification of your whole client application and make further size savings.

Enable zlib compression

Diffusion clients use zlib to support message compression. Since Diffusion 6.1, zlib code for message decompression has been removed from the main JavaScript client library to reduce its size and separated out into `browserify-zlib-0.2.0.js`.

`browserify-zlib-0.2.0.js` is included in the `clients/js` directory of a Diffusion installation, or can be downloaded from JavaScript SDK downloads.

Include `browserify-zlib-0.2.0.js` for clients that want to make use of the client compression capability. This can be achieved at build time by using browserify to package the browserify-zlib npm module into the application library.

Clients will log out a warning at startup if `browserify-zlib-0.2.0.js` is not included. The client's initial connection request will set the per-message compression capability depending on whether it is included or not. This will indicate to the server whether messages should be compressed before they are sent to the client.

Modularized browser bundles

As well as the full client, modularized browser bundles are available to minimize page size.

This enables you to use the core bundle with minimal features, and additional bundles which can be loaded into the browser dynamically after the core bundle of the modular client.

There is also a `diffusion-worker.js` script which enables you to use web workers to share sessions across browser tabs.

The bundles are available in the `node_modules/diffusion/dist` directory of an installation with npm, or in the `clients/js` directory of an on-premise Diffusion installation.

See the JavaScript JavaScript API documentation for full details.

Use TypeScript definitions with the JavaScript client library:

If you got the JavaScript client library using NPM, the TypeScript definitions are included.

You can also download a TypeScript definition file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/js/diffusion-6.7.7.d.ts
```

The TypeScript file is also located in your Diffusion server installation:

```
diffusion_directory/clients/js
```

Include the TypeScript definition file in your IDE project to use the TypeScript definitions when developing a JavaScript client for Diffusion.

Use with webpack

The JavaScript npm client supports the use of webpack and has been tested with webpack 4.16.2.

Targeting Node.js with webpack

When targeting Node.js with webpack, to avoid a dependency issue, add this line to `webpack.config.js`:

```
plugins: [ new webpack.IgnorePlugin(/vertx/) ]
```

Ensure that at the top of `webpack.config.js` you have:

```
const webpack = require('webpack');
```

Capabilities

To see the full list of capabilities supported by the JavaScript API, see Feature support in the Diffusion API on page 32.

Support

For information about the browsers supported by the Diffusion JavaScript client, see Browser support on page 39.

Table 24: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
JavaScript	es6 (TypeScript 1.8)	WebSocket HTTP (Polling XHR)

Resources

- Examples for the JavaScript API.
- JavaScript API documentation

Using

Promises

The Diffusion JavaScript API uses the Promises/A+ specification.

Views

The JavaScript API provides a view capability.

Use views to subscribe to multiple topics by using a topic selector and receive all the data from all topics in the selector set as a single structure when any of the topics are updated. If the topic selector matches a topic which is subsequently added or removed, the view is updated.

The following example shows views being used to present data from multiple topics as a single structure:

```
diffusion.connect({
    host        : 'localhost',
    port        : 8080,
    secure      : false,
    principal   : 'control',
    credentials : 'password'
}).then(function(session) {

    // Assuming a topic tree:
    //
    // scores
    //   |-- football
    //   |     |-- semi1
    //   |     |-- semi2
    //   |     |-- final
    //   |
    //   |-- tennis
    //         |-- semi1
    //         |-- semi2
    //         |-- final

    // Use a regular expression to create a view of the
 topics tracking the
    // scores during the finals for each sport.
    var view = session.view('?scores/.*/final');

    // Alternatively, we can use a topic set. Note that
 the topics do not need
    // to be under a common root, they may be anywhere
 within the topic tree.
    var view2 = session.view('#>scores/football/final////
>scores/tennis/final');

    // If any of the topics in the view change, display
 which topic changed
    // and its new value.
    view.on({
        update : function(value) {
            // Get and print the entire view structure.
            console.log('Update: ', JSON.stringify(value,
 undefined, 4));

            // Get individual topics. Returns a Buffer,
 which is automatically
            // converted to a String during
 concatenation, below.
            //
            // Note that the structure may not exist if
 the value has not been
            // updated.
            console.log('Football score: ' +
 value.scores.football.final);
            console.log('Tennis score  : ' +
 value.scores.tennis.final);

            // or ...
            // console.log('Football score: ' +
 value['scores']['football']['final']);
```

```
        }
    });

    // The structure can also be accessed outside the
 update event.
    console.log('Football score: ' +
 view.get().scores.football.final);
});
```

Regular expressions

The JavaScript client uses a different regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. It is possible that topic selectors that include complex or advanced regular expressions can behave differently on the client and on the Diffusion server.

For more information, see Regular expressions on page 49.

Apple

The Apple SDK is provided for iOS, OS X/macOS, and tvOS.

Get the Apple SDK for iOS:

Download the SDK from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/apple/diffusion-iphoneos-6.7.7.zip
```

The SDK file is also located in your Diffusion server installation:

```
diffusion_directory/clients/apple/diffusion-iphoneos-6.7.7.zip
```

Get the Apple SDK for OS X/macOS:

Download the SDK from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/apple/diffusion-macosx-6.7.7.zip
```

The SDK file is also located in your Diffusion server installation:

```
diffusion_directory/clients/apple/diffusion-macosx-6.7.7.zip
```

Get the Apple SDK for tvOS:

Download the SDK from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/apple/diffusion-appletvos-6.7.7.zip
```

The SDK file is also located in your Diffusion server installation:

```
diffusion_directory/clients/apple/diffusion-appletvos-6.7.7.zip
```

Capabilities

To see the full list of capabilities supported by the Apple API, see Feature support in the Diffusion API on page 32.

Support

Table 25: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
Apple for iOS	**Development environment** Xcode 12.4 (iOS 10.0 SDK) **Runtime support** Device architectures: armv7, armv7s, arm64 Simulator architectures: i386, x86_64	WebSocket
Apple for OS X/macOS	**Development environment** Xcode 12.4 (OS X/macOS 10.12 SDK) **Runtime support** Device architectures: x86_64	WebSocket
Apple for tvOS	**Development environment** Xcode 12.4 **Runtime support** Device architectures: arm64 Simulator architectures: x86_64	WebSocket

Resources

- Objective-C examples for the Apple API.
- Apple API documentation

Using

Applications in background state

Apple applications can be sent to the background. When this happens your application is notified by the `applicationDidEnterBackground` callback. Applications go into background state before being suspended.

Applications can be sent to the background or suspended at any time. We recommend that your Diffusion app saves its state – in particular, any topic subscriptions – as this state changes.

When your Diffusion app is sent to the background, we recommend the client closes its session with the Diffusion server. When the Diffusion app returns to the foreground, it can open a new client session with the Diffusion server and use the saved state to restore topic subscriptions.

For more information, see the Apple App Life Cycle documentation and Strategies for Handling App State Transitions.

Regular expressions

The Apple client uses a different regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. It is possible that topic selectors that include complex or advanced regular expressions can behave differently on the client and on the Diffusion server.

For more information, see Regular expressions on page 49.

Android

The Android API is bundled in a JAR file and is supported on Android KitKat and later.

To compile the Diffusion Android APK, add the following to the `build.gradle` file within your project:

```
apply plugin: 'com.android.application'
android {
  compileSdkVersion 27
  defaultConfig {
    multiDexEnabled true
    applicationId "com.pushtechnology.android.example"
    minSdkVersion 19
    targetSdkVersion 27
    versionCode 1
    versionName "1.0"
  }
buildTypes {
  release {
    minifyEnabled false
    proguardFiles getDefaultProguardFile('proguard-android.txt'),
 'proguard-rules.pro'
  }
}
compileOptions {
```

```
  coreLibraryDesugaringEnabled true
  sourceCompatibility JavaVersion.VERSION_1_8
  targetCompatibility JavaVersion.VERSION_1_8
  }
}
repositories {
  maven {
  url "https://download.pushtechnology.com/maven/"
  }
}
dependencies {
  coreLibraryDesugaring 'com.android.tools:desugar_jdk_libs:1.0.9'
  implementation 'org.slf4j:slf4j-api:1.7.21'
  implementation 'com.pushtechnology.diffusion:diffusion-android-
client:6.7.7'
}
```

Capabilities

To see the full list of capabilities supported by the Android API, see Feature support in the Diffusion API on page 32.

Support

Table 26: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
Android	API 19 / v4.4 / KitKat and later **Note:** Push Technology provides only best-effort support for Jelly Bean (API 16-18, v4.1-4.3).	WebSocket HTTP (polling)

Resources

- Java examples for the Android API.
- Android API documentation

Using

Considerations and capabilities that are specific to the Android API

Diffusion connections

Ensure that you use the asynchronous `open()` method with a callback. Using the synchronous `open()` method might open a connection on the same thread as the UI and cause a runtime exception. However, the synchronous `open()` method can be used in any thread that is not the UI thread.

Applications in background state

Android applications can be sent to the background and their activity stopped. When this happens your application is notified by the `onStop()` callback of the Android `Activity` class. An application's activity can be stopped when the user switches to another application, starts a new activity from within the application, or receives a phone call.

When your application's activity is stopped, we recommend that it saves its state locally – in particular, any topic subscriptions it has made – and closes its client session with the Diffusion server. When the Diffusion app returns to the foreground, open a new client session with the Diffusion server and use the saved state to restore topic subscriptions.

For more information, see the Android Activity Lifecycle documentation and Stopping and Restarting an Activity.

Writing good callbacks

The Android client library invokes callbacks using a thread from Diffusion thread pool. Callbacks for a particular session are called in order, one at a time. Consider the following when writing callbacks:

- Do not sleep or call blocking operations in a callback. If you do so, other pending callbacks for the session are delayed. If you must call a blocking operation, schedule it in a separate application thread.
- You can use the full Diffusion API to make other requests to the server. If you want to make many requests based on a single callback notification, be aware that Diffusion client flow control is managed differently in callback threads. Less throttling is applied and it is easier to overflow the servers by issuing many thousands of requests. If you have a lot of requests to make, it is better to schedule the work in an application thread.

Regular expressions

The Android client uses the same regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. There is no difference in how these regular expressions are evaluated in the Android client.

Java

The Java API is provided as a JAR file. It can be used with Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later).

Get the Java client libraries using Maven™:

Add the Push Technology public repository to your pom.xml file:

```
<repositories>
    <repository>
        <id>push-repository</id>
        <url>https://download.pushtechnology.com/maven/</url>
    </repository>
</repositories>
```

Declare the following dependency in your pom.xml file:

```
<dependency>
    <groupId>com.pushtechnology.diffusion</groupId>
    <artifactId>diffusion-client</artifactId>
    <version>6.7.7</version>
</dependency>
```

Get the Java client libraries using Gradle:

Add the Push Technology public repository to your build.gradle file:

```
repositories { maven { url "http://download.pushtechnology.com/
maven/" } }
```

Declare the following dependency in your build.gradle file:

```
compile 'com.pushtechnology.diffusion:diffusion-client:6.7.7'
```

Get the Java client libraries:

Download the JAR file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/java/diffusion-
client-6.7.7.jar
```

The ZIP file is also located in your Diffusion server installation:

```
diffusion_directory/clients/java/diffusion-client-6.7.7.jar
```

Capabilities

To see the full list of capabilities supported by the Java API, see Feature support in the Diffusion API on page 32.

Support

Table 27: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
Java	Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later) **Note:** The JVM must be HotSpot based. **Note:** The JVM must either have passed the Java TCK or be an official AdoptOpenJDK build.	WebSocket HTTP (Polling)

Resources

- Examples for the Java API.
- Java API documentation

Using

Certificates

Diffusion Java clients use certificates to validate the security of their connection to the Diffusion server. The client validates the certificate sent by the Diffusion server against the set of certificates trusted by the .

If the certificate sent by the Diffusion server cannot be validated against any certificates in the set trusted by the JDK, you receive an exception that contains the following message: `sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target`.

Diffusion is authenticated using the certificates provided by your certificate authority for the domain you host the Diffusion server on.

To ensure that the certificate is validated, set up a trust store for the client and add the appropriate certificates to that trust store:

1. Obtain the appropriate intermediate certificate from the certificate authority.
2. Use keytool to create a trust store for your client that includes this certificate.

   For more information, see https://docs.oracle.com/cd/E19509-01/820-3503/ggfka/index.html
3. Use system properties to add the trust store to your client.

   For example:

   ```
   System.setProperty("javax.net.ssl.trustStore",
    "truststore_name");
   ```

   Or at the command line:

   ```
   -Djavax.net.ssl.keyStore=path_to_truststore
   ```

Writing good callbacks

The Java client library invokes callbacks using a thread from Diffusion thread pool. Callbacks for a particular session are called in order, one at a time. Consider the following when writing callbacks:

- Do not sleep or call blocking operations in a callback. If you do so, other pending callbacks for the session are delayed. If you must call a blocking operation, schedule it in a separate application thread.
- You can use the full Diffusion API to make other requests to the server. If you want to make many requests based on a single callback notification, be aware that Diffusion client flow control is managed differently in callback threads. Less throttling is applied and it is easier to overflow the servers by issuing many thousands of requests. If you have a lot of requests to make, it is better to schedule the work in an application thread.

Regular expressions

The Java client uses the same regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. There is no difference in how these regular expressions are evaluated in the Java client.

.NET

The .NET API is provided as a package compatible with Microsoft .NET Standard 2.0.

Get the .NET SDK from NuGet:

Use the following .NET CLI command:

```
dotnet add package Diffusion.Client -v 6.7.7
```

Get the .NET SDK:

Download the .NET SDK files from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/dotnet/diffusion-dotnet-6.7.7.zip
```

These files are also located in your Diffusion server installation:

```
diffusion_directory/clients/dotnet
```

Capabilities

To see the full list of capabilities supported by the .NET API, see Feature support in the Diffusion API on page 32.

Support

Table 28: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
.NET	Microsoft .NET Standard 2.0	WebSocket

Resources

- Examples for the .NET API.
- .NET API documentation

Using

Certificates

Diffusion .NET clients use certificates to validate the security of their connection to the Diffusion server. The client validates the certificate sent by the Diffusion server against the set of certificates trusted by the .NET Framework.

If the certificate sent by the Diffusion server cannot be validated against any certificates in the set trusted by the .NET Framework, you must set up a trust store for the client and add the appropriate certificates to that trust store.

Diffusion is authenticated using the certificates provided by your certificate authority for the domain you host the Diffusion server on.

1. Obtain the appropriate intermediate certificate from the certificate authority.

2. Use the **Microsoft Management Console** to import the certificate into the **Trusted Root Certification Authorities** folder. For more information, see https://msdn.microsoft.com/en-us/library/aa738659(v=vs.110).aspx

Writing good callbacks

The .NET client library invokes callbacks using a single inbound thread. Callbacks for a particular session are called in order, one at a time. Consider the following when writing callbacks:

- Do not sleep or call blocking operations in a callback. If you do so, other pending callbacks for the session are delayed. If you must call a blocking operation, schedule it in a separate application thread.
- You can use the full Diffusion API to make other requests to the server. If you want to make many requests based on a single callback notification, be aware that Diffusion client flow control is managed differently in callback threads. Less throttling is applied and it is easier to overflow the servers by issuing many thousands of requests. If you have a lot of requests to make, it is better to schedule the work in an application thread.

Regular expressions

The .NET client uses a different regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. It is possible that topic selectors that include complex or advanced regular expressions can behave differently on the client and on the Diffusion server.

For more information, see Regular expressions on page 49.

C

The C client libraries are provided for Linux, Windows, and OS X/macOS.

Get the C client libraries for Linux:

Download the ZIP file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/c/diffusion-c-6.7.7.zip
```

The ZIP file is also located in your Diffusion server installation:

```
diffusion_directory/clients/c/diffusion-c-6.7.7.zip
```

Get the C client libraries for Windows:

Download the ZIP file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/c/diffusion-c-windows-6.7.7.zip
```

The ZIP file is also located in your Diffusion server installation:

```
diffusion_directory/clients/c/diffusion-c-windows-6.7.7.zip
```

Get the C client libraries for OS X/macOS:

Download the ZIP file from the following URL:

```
https://download.pushtechnology.com/clients/6.7.7/c/diffusion-c-osx-6.7.7.zip
```

The ZIP file is also located in your Diffusion server installation:

```
diffusion_directory/clients/c/diffusion-c-osx-6.7.7.zip
```

Capabilities

To see the full list of capabilities supported by the C API, see Feature support in the Diffusion API on page 32.

Support

Table 29: Supported platforms and transport protocols for the client libraries

Platform	Minimum supported versions	Supported transport protocols
C for Linux	Red Hat and CentOS, version 7.2 and later Ensure that you use a C99-capable compiler.	WebSocket
C for Windows	Visual C Compiler 2013 or later, Windows 7 or later	WebSocket
C for OS X/macOS	For building using GCC, use Xcode 8.0 or later	WebSocket

If you require libraries compiled on a different platform, this can be provided as an additional service by our Consulting Services team. Contact support@pushtechnology.com to discuss your requirements.

Resources

- Examples for the C API.
- C API documentation

Using

On Linux

The C libraries are provided compiled for 64-bit Linux in the file `diffusion-c-version.zip`. A dynamic library, `libdiffusion.so`, and a static library, `libdiffusion.a`, are available.

To use the C API on Linux ensure that the following dependencies are available on your development system:

- Perl Compatible Regular Expressions (PCRE) library, version 8.3 or later

  For more information, see http://pcre.org
- zLib library, version 1.2 or later

  For more information, see http://www.zlib.net

You can download these dependencies through your operating system's package manager.

The C client library statically links to APR version 1.6.1 with APR-util. Ensure that you set `APR_DECLARE_STATIC` and `APU_DECLARE_STATIC` before you use any APR includes. You can set these values in the following ways:

- By including `diffusion.h` before any APR includes. The `diffusion.h` file sets these values.
- As command-line flags.

For more information, see http://apr.apache.org

On Windows

The C library is provided as a static library compiled for 32-bit and 64-bit Windows in the file `diffusion-c-windows-version.zip`. This static library, `uci.lib`, is compiled with Visual C Compiler 2015 (version 140), which is shipped by default with Microsoft Visual Studio 2015. You must use this version of Visual C Compiler or later and use Windows 7 or later. Earlier versions are not supported.

Other Windows compilers, such as Clang and GCC, are not supported.

When compiling with Visual C Compiler 2015, define `/D WIN32` in the compiler settings.

To use the C API on Windows ensure that the following dependencies are available on your development system:

- Perl Compatible Regular Expressions (PCRE) library, version 8.3 or later

  For more information, see http://pcre.org
- zLib library, version 1.2 or later

  For more information, see http://www.zlib.net

We provide these dependencies in the `diffusion-c-windows-version.zip` file.

The C client library statically links to APR version 1.6.1 with APR-util. Ensure that you set `APR_DECLARE_STATIC` and `APU_DECLARE_STATIC` before you use any APR includes. You can set these values in the following ways:

- By including `diffusion.h` before any APR includes. The `diffusion.h` file sets these values.
- As command-line flags.

For more information, see http://apr.apache.org

On OS X/macOS

The C library is provided as a static library, `libdiffusion.a`, compiled for 64-bit OS X/macOS in the file `diffusion-c-osx-version.zip`.

To use the C API on OS X/macOS ensure that the following dependencies are available on your development system:

- Perl Compatible Regular Expressions (PCRE) library, version 8.3 or later

  For more information, see http://pcre.org
- zLib library, version 1.2 or later

  For more information, see http://www.zlib.net

You can download this dependencies using `brew`.

The C client library statically links to APR version 1.6.1 with APR-util. Ensure that you set APR_DECLARE_STATIC and APU_DECLARE_STATIC before you use any APR includes. You can set these values in the following ways:

- By including `diffusion.h` before any APR includes. The `diffusion.h` file sets these values.
- As command-line flags.

For more information, see http://apr.apache.org

For building using GCC, use Xcode 7.1 or later, which includes Apple LLVM.

Defining the structure of record topic data using XML

Data on record topics can be structured using metadata. Other Diffusion APIs provide builder methods you can use to define the metadata structure. The C API uses XML to define the structure of a record topic's metadata.

The following schema describes the structure of that XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://
www.w3.org/2001/XMLSchema">

  <xs:element name="field" type="field"/>

  <xs:element name="message" type="message"/>

  <xs:element name="record" type="record"/>

  <xs:complexType name="record">
    <xs:complexContent>
      <xs:extension base="node">
        <xs:sequence>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="record"/>
            <xs:element ref="field"/>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="node">
    <xs:sequence/>
    <xs:attribute name="name" type="xs:string"
 use="required"/>
    <xs:attribute name="multiplicity" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="field">
    <xs:complexContent>
      <xs:extension base="node">
        <xs:sequence/>
        <xs:attribute name="type" type="dataType"
 use="required"/>
        <xs:attribute name="default" type="xs:string"/>
        <xs:attribute name="scale" type="xs:integer"/>
        <xs:attribute name="allowsEmpty"
 type="xs:boolean"/>
        <xs:attribute name="customFieldHandlerClassName"
 type="xs:string"/>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="message">
    <xs:complexContent>
      <xs:extension base="record">
        <xs:sequence/>
        <xs:attribute name="topicDataType"
 type="topicDataType"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="dataType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="integerString"/>
      <xs:enumeration value="string"/>
      <xs:enumeration value="customString"/>
      <xs:enumeration value="decimalString"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="topicDataType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="record"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Threading model

The C API is not thread-safe. Session and their derived artifacts must belong to a single thread or only be acted upon by a single thread at any time.

Internally, the C client creates threads for managing the connection to the Diffusion server. All callbacks into user-defined code are synchronous and it usually the case that these must execute as quickly as possible. If this code runs for a non-trivial amount of time, ensure that it hands off work to your own threads.

It is safe to send messages while processing callbacks, as outbound messages are queued and are sent as soon as possible.

Ensure that callbacks do not alter the session as this can lead to undefined behavior. This includes calling functions such as `session_close()` from the session state change callback.

Always call `session_close()` and `session_free()` from the same thread that created the session with `session_create()` or `session_create_async()`. This allows the threads to be joined and reaped correctly, and is a requirement of the APR library which the C API relies on.

Regular expressions

The C client uses a different regular expression engine to the Diffusion server. Some regular expressions in topic selectors are evaluated on the client and others on the Diffusion server. It is possible that topic selectors that include complex or advanced regular expressions can behave differently on the client and on the Diffusion server.

For more information, see Regular expressions on page 49.

Python

The Python API can be installed using `pip` and is also included in the server installation.

Install with pip:

```
pip install diffusion
```

To install a specific version:

```
pip install diffusion==6.7.7
```

You can also download the SDK as a zipped WHL package:

```
https://download.pushtechnology.com/clients/6.7.7/python/diffusion-python-6.7.7.zip
```

If installing from this downloaded wheel, you will need to download a platform-specific CBOR implementation as well.

It will be one of the following:

OSX:

```
https://download.pushtechnology.com/clients/6.7.7/python/diffusion-python-cbor-osx-6.7.7.zip
```

Windows:

```
https://download.pushtechnology.com/clients/6.7.7/python/diffusion-python-cbor-windows-6.7.7.zip
```

Linux:

```
https://download.pushtechnology.com/clients/6.7.7/python/diffusion-python-cbor-linux-6.7.7.zip
```

Extract both the client and CBOR zip files.

You can then install with the command:

```
pip install {location of the client wheel} --find-links {directory
 containing unzipped CBOR}
```

Supported interpreters:

- CPython 3.7.8 (or later patch release)
- CPython 3.8.6 (or later patch release)
- CPython 3.9.0 (or later patch release)

Supported platforms:

- MacOS 10.13-11.5
- Windows Intel 64-bit
- Any Linux supported by the ManyLinux 2010/2014] binary wheel standard.

Future releases will be tested against additional interpreters.

We recommend using the newest available patch release.

Resources

- Examples for the Python API.
- Python API documentation

Connecting to the Diffusion server

One of the first actions your Diffusion client takes is to connect to the Diffusion server. Clients connect to the Diffusion server by opening a session. A session represents a logical context between a client and the Diffusion server. All interactions with the Diffusion server happen within a session.

Sessions

The act of opening the session establishes a connection with the the Diffusion server. When a session is opened it is assigned a unique session identifier by the Diffusion server, which identifies the session even if it becomes connected to another server.

The session does not receive input from the Diffusion server until it is started, but can be used to obtain features and perform certain setup actions before it is started.

Session state

The session between a client and the Diffusion server can be in one of a number of states.

The following diagram shows the session state model:

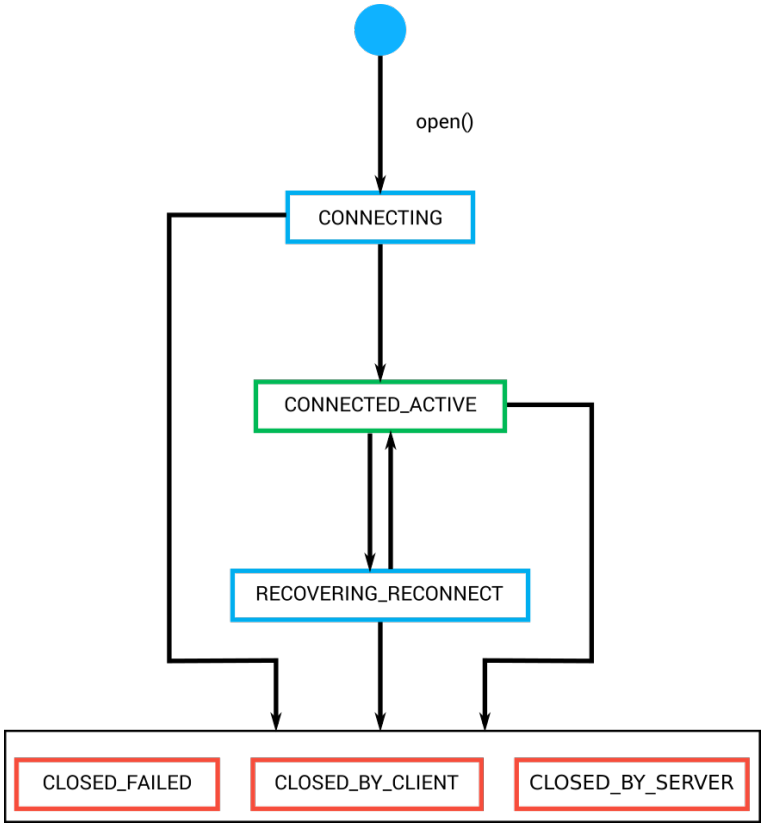


Figure 17: Session state model

`CONNECTING`

The client session is in this state while it attempts to connect to the Diffusion server. If the connection attempt is successful, the session changes to `CONNECTED_ACTIVE` state. If the connection is not successful, the session changes to one of the closed states.

`CONNECTED_ACTIVE`

The client session is in this state while it is connected to the Diffusion server. The session spends the majority of its lifetime in this state. If the session becomes disconnected and reconnect is enabled, the session changes to `RECOVERING_CONNECT` state. If the session closes, it changes to one of the closed states.

`RECOVERING_CONNECT`

The client session is in this state while it attempts to reconnect to the Diffusion server after a disconnection. If the reconnection attempt is successful, the session changes back to `CONNECTED_ACTIVE` state. If the reconnection attempt is not successful, the session changes to one of the closed states.

`CLOSED_BY_CLIENT`

The client session is in this state when it is closed by the client. If a session is in closed state, it cannot be reopened. A new session must be established.

`CLOSED_BY_SERVER`

The client session is in this state when it is closed by the Diffusion server. If a session is in closed state, it cannot be reopened. A new session must be established.

`CLOSED_FAILED`

The client session is in this state when it is closed for any reason other than a close by the client or by the Diffusion server. If a session is in closed state, it cannot be reopened. A new session must be established.

For more information, see Managing your session on page 192.

Session properties

When you connect to the Diffusion server by opening a session, the session is assigned a set of properties. These properties are assigned by either the Diffusion server or an authentication handler and can be used by clients to filter the set of connected client sessions to take actions on.

A client can propose session properties. The proposed session properties are passed to the authentication handler which can accept the properties, modify the properties, or ignore them entirely.

For more information, see Session properties on page 220.

Session roles

When a session authenticates with the Diffusion server, the session is assigned a set of roles. These roles are assigned by either the Diffusion server or an authentication handler and define the set of permissions a client session has.

For more information, see Role-based authorization on page 138.

Connecting basics

To make a connection to the Diffusion server the client must specify the host name and port number of the Diffusion server, the transport to use to connect, and whether that connection is secure.

The Diffusion API is an asynchronous API. As such, the client APIs for all languages provide asynchronous connect methods.

A subset of the Diffusion APIs also provide synchronous connect methods: the Android, Java, .NET, and C APIs. In the following sections, all examples for these APIs are synchronous for simplicity. For asynchronous examples for these APIs, see Asynchronous connections on page 190.

Connection parameters

host

The host name or IP address of the system on which the Diffusion server is located.

port

The port on which the Diffusion server accepts connections from clients using the Diffusion API. You can configure which ports to provide connectors for in the `Connectors.xml` configuration file. For more information, see Connectors.xml on page 438.

transport

The transport used to make the connection. For example, WebSocket (`ws`). The transports your client can use to make a connection depend on the client library capabilities. For more information, see Platform support for the Diffusion API libraries on page 30.

secure

Whether the connection is made over SSL.

Connecting

In JavaScript, Android and Java, you can define each of these parameters individually:

JavaScript

```
diffusion.connect({
    host : 'host_name',
    port : 'port', // If not specified, port defaults to 80 for
 standard connections or 443 for secure connections
    transports : 'transport', // If not specified, transports
 defaults to 'WS' and the client uses a WebSocket connection
    secure : false  // If not specified, secure defaults to false
}).then(function(session) { ... } );
```

Java and Android

```
final Session session = Diffusion
    .sessions()
    .serverHost("host_name")
    // If no port is specified, the port defaults to 80 for standard
 connections or 443 for secure connections
    .serverPort(port)
```

```
    // If no transports are specified, the connection defaults to use
 the WebSocket transport
    .transports(transport)
 // If not specified, secure transport defaults to false
    .secureTransport(false)
    .open();
```

In Apple, Android, Java, .NET and C, composite the host, port, transport, and whether the connection is secure into a single URL-style string of the following form: `transport[s]://host:port`.

For example, `ws://diffusion.example.com:8080`.

Use this URL to open the connection to the Diffusion server:

.NET

```
var session = Diffusion.Sessions.Open( "url" );
```

Java and Android

```
Session session = Diffusion.sessions().open("url");
```

C

```
SESSION_T *session = session_create(url, NULL, NULL,
 &session_listener, NULL, NULL);
```

Apple

```
[PTDiffusionSession openWithURL:[NSURL URLWithString:@"url"]
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Connecting with multiple transports

In JavaScript, you can specify a list of transports. The client uses these transports to provide a *transport cascading* capability.

JavaScript

```
diffusion.connect({
    host : 'host_name',
    transports : ['transport','transport','transport']
}).then(function(session) { ... } );
```

Java and Android

```
final Session session = Diffusion
    .sessions()
    .serverHost("host_name")
    .serverPort(port)
    .transports(transport, transport, transport)
    .open();
```

1. The client attempts to connect using the first transport listed.
2. If the connection is unsuccessful, the client attempts to connect using the next transport listed.
3. This continues until the one of the following events happens:
    - The client makes a connection
    - The client has attempted to make a connection using every listed transport. If this happens, the connection fails.

You can use specify that a client attempt to connect with a transport more than once. This enables you to define retry behavior. For example:

JavaScript

```
transports: ['WS', 'XHR', 'WS']
```

Java and Android

```
.transports(WEBSOCKET, WEBSOCKET, WEBSOCKET)
```

Transport cascading is useful to specify in your clients as it enables them to connect from many different environments. Factors such as the firewall in use, your end-user's mobile provider, or your end-user's browser can affect which transports can successfully make a connection to the Diffusion server.

Asynchronous connections

All Diffusion APIs can connect asynchronously to the Diffusion server:

JavaScript

```
diffusion.connect({
    host : 'host_name',
    port : 'port'
}).then(function(session) { ... } );
```

.NET

```
// Define a callback that implements ISessionOpenCallback and pass
 this to the open method
Diffusion.Sessions.Open("url", callback );
```

Java and Android

```
// openAsync returns a CompletableFuture that completes with a
 Session when a response is received from the server
Diffusion.sessions().openAsync("url").thenApply(session -> { ... });
```

C

```
/*
 * Asynchronous connections have callbacks for notifying that
 * a connection has been made, or that an error occurred.
 */
 SESSION_CREATE_CALLBACK_T *callbacks = calloc(1,
 sizeof(SESSION_CREATE_CALLBACK_T));
 callbacks->on_connected = &on_connected;
 callbacks->on_error = &on_error;

 session_create_async(url, principal, credentials, &session_listener,
 reconnection_strategy, callbacks, &error);
```

Apple

```
// Excluding the port from the URL defaults to 80, or 443 for secure
 connections
[PTDiffusionSession openWithURL:[NSURL URLWithString:@"url"]
              completionHandler:^(PTDiffusionSession * newSession,
 NSError * error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Synchronous connections

The following APIs can connect synchronously to the Diffusion server:

.NET

```
var session = Diffusion.Sessions.Open( "url" );
```

Java and Android

```
Session session = Diffusion.sessions().open("url");
```

C

```
```

```
SESSION_T *session = session_create(url, NULL, NULL,
 &session_listener, NULL, NULL);
```

When connecting to the Diffusion server using the Android API, prefer the asynchronous `open()` method with a callback. Using the synchronous `open()` method might open a connection on the same thread as the UI and cause a runtime exception. However, the synchronous `open()` method can be used in any thread that is not the UI thread.

Managing your session

When your client has opened a session with the Diffusion server, you can listen for session events to be notified when the session state changes. For more information about session states, see Session state on page 186.

JavaScript

In JavaScript, listen for the following events on the session:

- `disconnect`: The session has lost connection to the Diffusion server.

  The session state changes from `CONNECTED_ACTIVE` to `RECOVERING_RECONNECT`. This event is only emitted if reconnect is enabled.
- `reconnect`: The session has re-established connection to the Diffusion server.

  The session state changes from `RECOVERING_RECONNECT` to `CONNECTED_ACTIVE`.
- `close`: The session has closed. The provided close reason indicates whether this was caused by the client, the Diffusion server, a failure to connect, or an error.

  The session state changes to one of `CLOSED_FAILED`, `CLOSED_BY_SERVER`, or `CLOSED_BY_CLIENT`.
- `error`: A session error occurs.

JavaScript

```
session.on('disconnect', function() {
    console.log('Lost connection to the server.');
});
session.on('reconnect', function() {
    console.log('Reconnected to the session on the server.');
});
session.on('close', function() {
    console.log('Session is closed.');
});
session.on('error', function() {
    console.log('A session error occurred.');
});
```

Apple

In Apple, the following boolean properties are available on the states that are broadcast through the default notification center for the application process and posted on the main dispatch queue:

- `isConnected`: If true, the state is equivalent to the `CONNECTED_ACTIVE` state.
- `isRecovering`: If true, the state is equivalent to the `RECOVERING_RECONNECT` state.
- `isClosed`: If true, the state is one of `CLOSED_FAILED`, `CLOSED_BY_SERVER`, or `CLOSED_BY_CLIENT`.

The broadcast includes both the old state and new state of the session. It also includes an error property that is `nil` unless the session closure was caused by a failure.

Apple

```
NSNotificationCenter * nc = [NSNotificationCenter defaultCenter];
[nc addObserverForName:PTDiffusionSessionStateDidChangeNotification
                object:session
                 queue:nil
            usingBlock:^(NSNotification * note)
{
    PTDiffusionSessionStateChange * change =
 note.userInfo[PTDiffusionSessionStateChangeUserInfoKey];
    NSLog(@"Session state change: %@", change);
}];
```

Other SDKs

In Android, Java, .NET, and C listen for changes to the session state. The listener provides both the old state and new state of the session. The states provided are those listed in the session state diagram. For more information, see Session state on page 186.

.NET

```
// Add the listener to the session factory you will use to create the
 session
var sessionFactory =
 Diffusion.Sessions.SessionStateChangedHandler( ( sender, args ) => {

    Console.WriteLine( "Session state changed from " +
 args.OldState.ToString() + " to " + args.NewState.ToString() );

} );
```

Java and Android

```
// Add the listener to the session
session.addListener(new Listener() {
    @Override
    public void onSessionStateChanged(Session session, State
 oldState, State newState) {

        System.out.println("Session state changed from " +
 oldState.toString() + " to " + newState.toString());

    }
});
```

C

```
// Define a session listener
static void
on_session_state_changed(SESSION_T *session,
        const SESSION_STATE_T old_state,
        const SESSION_STATE_T new_state)
{
        printf("Session state changed from %s (%d) to %s (%d)\n",
               session_state_as_string(old_state), old_state,
```

```
                session_state_as_string(new_state), new_state);
}

// ...

    // Use the session listener when opening your session
    SESSION_LISTENER_T session_listener = { 0 };
            session_listener.on_state_changed =
 &on_session_state_changed;

    session_create_async(url, principal, credentials,
 &session_listener, &reconnection_strategy, callbacks, &error);
```

Connecting securely

A Diffusion client can make secure connections to the Diffusion server over TLS. All supported transports can connect securely.

To connect securely do one of the following:

- In JavaScript, set the `secure` parameter to `true`
- In Android and Java, when specifying parameters individually, pass `true` to the `secureTransport()` method.
- If using a URL to connect, insert an “s” after the transport value in the `url` parameter. For example, `wss://diffusion.example.com:443`.

Configure the SSL context or behavior

A secure connection to the Diffusion server uses SSL to secure the communication.

When connecting over SSL, you might need to configure SSL.

- In JavaScript, the SSL context is provided by the browser.
- In Android, Java, and .NET, you can provide an SSL context when creating the session.
- In Apple, you can use the `sslOptions` property to provide a dictionary of values that specify the SSL behavior. For more information, see the `CFStreamConstants` documentation.

.NET

```
var session =
 Diffusion.Sessions.SslContext(ssl_context).Open( "secure_url" );
```

Java and Android

```
Session session =
 Diffusion.sessions().sslContext(ssl_context).open("secure_url");
```

If no SSL context or behavior is specified, the client uses the default context or configuration.

Validating server-side certificates

Diffusion clients that connect over a secure transport use certificates to validate the security of their connection to the Diffusion server. These certificates are validated against any certificates in the set trusted by the framework, runtime, or platform that the client library runs on.

If the client does not trust the certificate provided by a CA, you can configure the client to add certificates to its trust store:

- For Java, see Certificates on page 177
- For .NET, see Certificates on page 179

You can also write a trust manager that explicitly allows the CA's certificates.

Disabling certificate validation on the client

You can disable client validation of the server-side certificates.

> **Note:** We do not recommend disabling this validation on your production clients. However, it can be useful for testing.

Certificates can only be strictly validated if they have been issue by an appropriate Certificate Authority (CA) and if the CA's certificates are also known to your client.

Since certificates are specific to the domain name that the server is deployed on, Diffusion ships with demo certificates and these cannot be strictly validated. To test against a server with demo certificates, disable client-side SSL certificate validation as shown in the following examples:

Java and Android

```
TrustManager tm = new X509TrustManager() {
    public void checkClientTrusted(X509Certificate[] chain, String
 authType) throws CertificateException {
    }

    public void checkServerTrusted(X509Certificate[] chain, String
 authType) throws CertificateException {
    }

    public X509Certificate[] getAcceptedIssuers()
    {
         return new X509Certificate[0];
    }
};

final SSLContext context = SSLContext.getInstance("TLS");
context.init( null, new TrustManager[] { tm }, null );

Session session =
 Diffusion.sessions().sslContext(context).open("secure_url");
```

C

```
//Set this environmental variable
 DIFFUSION_TRUST_SELF_SIGNED_CERTS=1
```

Apple

```
// Create a session configuration with non-standard SSL options...
PTDiffusionMutableSessionConfiguration *const configuration =
    [PTDiffusionMutableSessionConfiguration new];
configuration.sslOptions = @{
    (__bridge id)kCFStreamSSLValidatesCertificateChain
```

```
        : (__bridge id)kCFBooleanFalse
};

// Use the configuration to open a new session...
[PTDiffusionSession openWithURL:[NSURL URLWithString:@"wss://
TestServer"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Connecting to the Diffusion server with a security principal and credentials

The Diffusion server can accept anonymous connections. However, if your clients specify a security principal (for example, a username) and its associated credentials (for example, a password) when they connect, these client sessions can be authenticated and authorized in a more granular way.

Authentication parameters

principal

A string that contains the name of the principal or identity that is connecting to the Diffusion server. If a value is not specified when connecting, the principal defaults to ANONYMOUS.

credentials

Credentials are a piece of information that authenticates the principal. This can be empty or contain a password, a cryptographic key, an image, or any other piece of information.

If you connect to the Diffusion server using a principal and credentials, connect over SSL to ensure that these details are encrypted.

Connecting using any type of credentials

In JavaScript and C the method that opens a connection to the Diffusion server takes `principal` and `credentials` as parameters:

JavaScript

```
diffusion.connect({
    host : 'host_name',
    port : 'port',
 principal: 'principal',
    credentials: 'credentials'
});
```

C

```
SESSION_T *session = session_create(url, principal, credentials,
 &session_listener, NULL, NULL);
```

Python

```
# Diffusion server connection information; same for both sessions
# adjust as needed for the server used in practice
server_url = "ws://localhost:8080"
principal = "admin"
credentials = diffusion.Credentials("password")

# creating the session
async with diffusion.Session(
    url=server_url, principal=principal, credentials=credentials
) as session:
    _ = session
    pass
```

Any form of credentials can be wrapped in a credentials object. This can be empty or contain a password, a cryptographic key, an image, or any other piece of information. The authentication handler is responsible for interpreting the bytes.

In the Apple, Android, Java, and .NET API specify the credentials as a credentials object. The principal and credentials are specified when configuring the session before opening it:

Apple

```
PTDiffusionCredentials *const credentials =
    [[PTDiffusionCredentials alloc] initWithData:data];

PTDiffusionSessionConfiguration *const configuration =
    [[PTDiffusionSessionConfiguration alloc]
 initWithPrincipal:@"principal"

 credentials:credentials];

[PTDiffusionSession openWithURL:[NSURL URLWithString:@"wss://
push.example.com"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Java and Android

```
Session session = Diffusion.sessions()
   .principal("principal")
   .credentials("credentials")
   .open("url");
```

.NET

```
var session = Diffusion.Sessions
   .Principal("principal")
   .Credentials("credentials")
   .Open( "url" );
```

Connecting using a string password as credentials

A string password is the most commonly used type of credentials. The Apple, Android, Java, and .NET API provide a convenience method that enables you to specify credentials as a string password. The principal and credentials are specified when configuring the session before opening it:

.NET

```
var session = Diffusion.Sessions
   .Principal("principal")
   .Password("credentials")
   .Open( "url" );
```

Java and Android

```
Session session = Diffusion.sessions()
   .principal("principal")
   .password("credentials")
   .open("url");
```

Apple

```
// Create a credentials object encapsulating a string password.
PTDiffusionCredentials *const credentials =
    [[PTDiffusionCredentials alloc] initWithPassword:@"password"];

PTDiffusionSessionConfiguration *const configuration =
    [[PTDiffusionSessionConfiguration alloc]
 initWithPrincipal:@"principal"

 credentials:credentials];

[PTDiffusionSession openWithURL:[NSURL URLWithString:@"url"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Connecting using a byte array as credentials

The Android, Java, and .NET API provide a convenience method that enables you to specify credentials as a byte array. The principal and credentials are specified when configuring the session before opening it:

.NET

```
var session = Diffusion.Sessions
   .Principal("principal")
   .CustomCredentials(byte_credentials)
   .Open( "url" );
```

Java and Android

```
Session session = Diffusion.sessions()
   .principal("principal")
   .customCredentials(byte_credentials)
   .open("url");
```

Changing the principal and credentials a session uses

The client session can change the principal and credentials it uses to connect to the Diffusion server at any time. For more information, see Change the security principal and credentials associated with your client session on page 219.

Connecting through an HTTP proxy

Clients can connect to the Diffusion server through an HTTP proxy by using the HTTP `CONNECT` verb to create the connection and tunneling any of the supported transports through that connection.

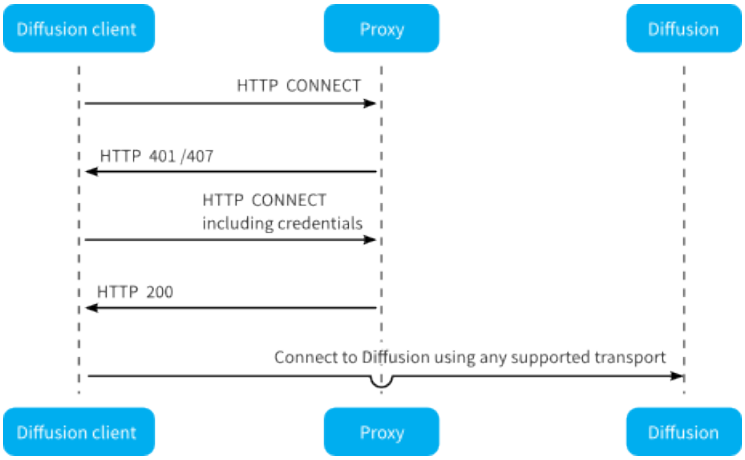


Figure 18: Flow of requests and responses when connecting to Diffusion through a proxy.

Apple, Android, Java, and .NET clients can connect to the Diffusion server through an HTTP proxy by specifying additional information on connection.

With no authentication at the proxy

When creating your session, add an HTTP proxy to the session by passing in the host and port number of the proxy.

JavaScript

```
Diffusion.sessions().httpProxy(host, port)
```

.NET

```
var session = Diffusion.Sessions
   .HttpProxy( host, port )
   .Open( diffusionUrl );
```

Apple

```
// Create a mutable session configuration.
PTDiffusionMutableSessionConfiguration *const configuration =
    [PTDiffusionMutableSessionConfiguration new];

// Create an unauthenticated HTTP proxy configuration.
PTDiffusionHTTPProxyConfiguration *const proxyConfiguration =
    [[PTDiffusionHTTPProxyConfiguration alloc] initWithHost:@"proxy"
                                                       port:80];

// Specify the proxy configuration.
configuration.httpProxyConfiguration = proxyConfiguration;

// Open the session, specifying this configuration.
[PTDiffusionSession openWithURL:[NSURL URLWithString:@"wss://
push.example.com"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

With basic authentication at the proxy

If the proxy requires basic authentication, the client can use the implementation in the Diffusion API to authenticate.

When creating your session, add an HTTP proxy to the session by passing in the host and port number of the proxy and a proxy authentication object that provides the challenge handler for basic authentication.

.NET

```
var clientAuth = Diffusion.ProxyAuthentication.Basic( username,
 password );
var session = Diffusion.Sessions
   .HttpProxy( host, port, clientAuth )
   .Open( diffusionUrl );
```

Java and Android

```
HTTPProxyAuthentication auth =
 Diffusion.proxyAuthentication().basic(username, password);
Diffusion.sessions().httpProxy(host, port, auth);
```

Apple

```
// Create a mutable session configuration.
PTDiffusionMutableSessionConfiguration *const configuration =
    [PTDiffusionMutableSessionConfiguration new];

// Create an authentication provider for the HTTP proxy.
const id<PTDiffusionHTTPAuthentication> authentication =
    [[PTDiffusionBasicHTTPProxyAuthentication alloc]
 initWithUsername:@"user"

 password:@"pass"];

// Create an authenticated HTTP proxy configuration using the
 provider.
PTDiffusionHTTPProxyConfiguration *const proxyConfiguration =
    [[PTDiffusionHTTPProxyConfiguration alloc] initWithHost:@"proxy"
                                                       port:80

 authentication:authentication];

// Specify the proxy configuration.
configuration.httpProxyConfiguration = proxyConfiguration;

// Open the session, specifying this configuration.
[PTDiffusionSession openWithURL:[NSURL URLWithString:@"wss://
push.example.com"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

With another form of authentication at the proxy

If the proxy requires another form of authentication, the client can implement a challenge handler that the client uses to authenticate.

Implement the `HTTPProxyAuthentication` interface to provide a challenge handler that can handle the type of authentication your proxy uses. When creating your session, add an HTTP proxy to the session by passing in the host and port number of the proxy and a proxy authentication object that provides your challenge handler.

Note: The proxy authentication mechanism is separate from the client authentication mechanism and is transparent to the Diffusion server.

Connecting through a load balancer

Connections between Diffusion clients and Diffusion servers can be routed through a load balancer. Some clients can pass additional information to a load balancer in the request path of their URL.

Supported in: JavaScript, Apple, Android, C and Java APIs

An additional request path can be specified to define the connection URL context.

JavaScript

```
diffusion.connect({
    host : 'host_name',
    port : 'port',
    transports : 'transport',
    secure : false,
    path: '/path/diffusion'
}).then(function(session) { ... } );
```

Java and Android

```
final Session session = Diffusion
    .sessions()
    .serverHost("host_name")
    .serverPort(port)
    .transports(transport)
    .secureTransport(false)
    .requestPath("/path/diffusion");
    .open();
```

Apple

```
NSString *const host = @"host";
NSString *const additionalInformationAsPath = @"path";

// Formulate the URL string for connection via secure WebSocket,
// appending the required '/diffusion' suffix.
NSString *const url = [NSString stringWithFormat:@"wss://%@/%@/
diffusion",
    host, additionalInformationAsPath];

// Open the session, using the formulated URL.
[PTDiffusionSession openWithURL:[NSURL URLWithString:url]
             completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

Specify a request path that begins with / and ends with /diffusion. The default value is /diffusion.

Load balancer configuration

Connections between Diffusion clients and Diffusion servers have specific requirements. If your load balancer handles Diffusion connections incorrectly, for example by routing subsequent client requests to different backend Diffusion servers, this can cause problems for your solution.

For more information about how to configure your load balancers to work with Diffusion, see Load balancers on page 625.

Reconnect to the Diffusion server

When clients connect to the Diffusion server over unreliable networks these connections can be lost. Clients can attempt to reconnect to the Diffusion server after they lose connection.

Diffusion keeps client sessions in the DISCONNECTED state for a period of time, during which the client can reconnect to the same session. The length of time the Diffusion server keeps a client session in the DISCONNECTED state for is configured for the connector that the client uses. For more information, see Configuring connectors on page 436.

Configuring reconnection on the client

Clients have reconnection enabled by default.

You can configure a reconnection timeout that restricts the amount of time the client can be disconnected and still reconnect to its session on the Diffusion server. The period of time that the Diffusion server keeps the session available for reconnect is the lowest of the following values:

- The reconnection timeout configured by the client when it creates its session
- The reconnection timeout configured on the Diffusion server for the connector that the client connects on

When the reconnection timeout period configured by the client ends, the client stops attempting to reconnect and closes its session.

JavaScript

```
diffusion.connect({
    host : 'url',
    reconnect : {
        // Specify the timeout in milliseconds
        timeout : reconnection_time
  }
    })
```

.NET

```
var session = Diffusion.Sessions
// Specify the timeout in milliseconds
    .ReconnectionTimeout(reconnection_time)
 .Open("url");
```

Java and Android

```
final Session session = Diffusion
    .sessions()
    // Specify the timeout in milliseconds
    .reconnectionTimeout(reconnection_time)
    .open("url");
```

C

```
reconnection_strategy_set_timeout(&reconnection_strategy,
 reconnection_time);
SESSION_T *session = session_create(url, NULL, NULL, NULL,
 &reconnection_strategy, NULL);
```

Apple

```
PTDiffusionMutableSessionConfiguration *const configuration =
    [PTDiffusionMutableSessionConfiguration new];

// Specify the timeout in seconds
configuration.reconnectionTimeout = @10;

[PTDiffusionSession openWithURL:[NSURL URLWithString:@"url"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
 {
     // Check error is `nil`, then use session as required.
     // Ensure to maintain a strong reference to the session beyond
 the lifetime
     // of this callback, for example by assigning it to an instance
 variable.
 }];
```

Set the value of the reconnection timeout to zero to disable reconnection. If no reconnection timeout is specified, a default of 60 seconds (60000 ms) is used.

You can also define your own custom reconnection behavior using reconnection strategies. For more information, see Specifying a reconnection strategy on page 206.

If no custom reconnection strategy is defined, the client attempts to reconnect at five second intervals until the reconnection timeout is reached.

Reliable reconnection

If a client loses connection to the Diffusion server, data sent between the client and the Diffusion server in either direction might be lost in transmission. If this happens and the client reconnects to its session on the Diffusion server, lost data might cause the client state or topic data to be incorrect.

To prevent any data being lost, the reconnection process re-synchronizes the streams of messages from the client session to the Diffusion server and from the Diffusion server to the client session. When reconnecting, the client notifies the Diffusion server of the last message received and the earliest message it can send again. The Diffusion server resends any missing messages and instructs the client session to resume from the appropriate message.

To be able to send messages again, the Diffusion server maintains a recovery buffer of sent messages. Some types of client also maintain a recovery buffer of sent messages that can be sent again if necessary.

If a message has been lost and is no longer present in the recovery buffer, the server will abort the reconnection. If reconnection succeeds, delivery of all messages is assured.

Configuring the recovery buffer on the client

All Diffusion clients can retain a buffer of messages that they have sent to the Diffusion server. If messages from the client are lost in transmission during a disconnection and subsequent reconnection, the client can resend the missing messages to the Diffusion server.

In the Apple, Java, Android and .NET APIs, you can configure the size of this buffer, in messages, when creating your session on the Diffusion server:

.NET

```
var session =
 Diffusion.Sessions.RecoveryBufferSize( number_of_messages ).Open("url");
```

Java and Android

```
final Session session = Diffusion
    .sessions()
    .recoveryBufferSize(number_of_messages)
    .open("url");
```

Apple

```
PTDiffusionMutableSessionConfiguration *const configuration =
    [PTDiffusionMutableSessionConfiguration new];

configuration.recoveryBufferSize = 1000;

[PTDiffusionSession openWithURL:[NSURL URLWithString:@"url"]
                  configuration:configuration
              completionHandler:^(PTDiffusionSession *session,
 NSError *error)
{
    // Check error is `nil`, then use session as required.
    // Ensure to maintain a strong reference to the session beyond
 the lifetime
    // of this callback, for example by assigning it to an instance
 variable.
}];
```

The default size of the recovery buffer is 128 messages.

The larger this buffer is, the greater the chance of successful reconnection. However, a larger buffer of messages increases the memory footprint of a client.

Configuring the recovery buffer on the Diffusion server

The recovery buffers on the Diffusion server can be configured on a per-connector basis in the `Connectors.xml` configuration file. For more information, see Configuring connectors on page 436.

Related concepts

Session reconnection on page 505
You can configure the session reconnection feature by configuring the connectors at the Diffusion server to keep the client session in a disconnected state for a period before closing the session.

Detecting connection problems

A client can automatically detect if there are problems with its connection to the Diffusion server and take action to handle any disconnection.

When a client detects that it has become disconnected from the Diffusion server, the session state changes from CONNECTED to one of the following states:

- If reconnection is enabled at the client and at the Diffusion server, the session state changes to RECOVERING.
- If reconnection is not enabled, the session state changes to DISCONNECTED.

The client can detect that it has become disconnected from the Diffusion server using the following methods:

Monitoring the connection activity

The client automatically monitors the activity between the client and the Diffusion server and uses this information to quickly discover any connection problems.

Using TCP state

Depending on the transport the client uses to connect to the Diffusion server, the client can use the TCP state to detect whether to change its state from CONNECTED to one of RECOVERING or DISCONNECTED.

- WebSocket: The client uses the TCP state to detect whether to trigger a state change.
- HTTP Polling: The client uses the TCP state at certain points during an HTTP request to detect whether to trigger a state change.

Specifying a reconnection strategy

Reconnection behavior can be configured using custom reconnection strategies.

The reconnection behavior of a client session can be configured using reconnection strategies. A reconnection strategy is applied when the session enters the RECOVERING_RECONNECT state, enabling the session to attempt to reconnect and recover its previous state.

Reconnection can only succeed if the client session is still available on the Diffusion server. The maximum time that the Diffusion server keeps client sessions in the DISCONNECTED state before closing them can be configured using the `Connectors.xml` configuration file. For more information, see Configuring connectors on page 436.

Individual client sessions can request a shorter reconnection timeout for their sessions or request to disable reconnection when they first connect to the Diffusion server

Examples

JavaScript

```
// When establishing a session, it is possible to specify whether
 reconnection
// should be attempted in the event of an unexpected disconnection.
 This allows
// the session to recover its previous state.

// Set the maximum amount of time we'll try and reconnect for to 10
 minutes
var maximumTimeoutDuration = 1000 * 60 * 10;

// Set the maximum interval between reconnect attempts to 60 seconds
var maximumAttemptInterval = 1000 * 60;

// Set an upper limit to the number of times we'll try to reconnect
 for
var maximumAttempts = 25;

// Count the number of reconnection attempts we've made
var attempts = 0;

// Create a reconnection strategy that applies an exponential back-
off
// The strategy will be called with two arguments, start & abort.
 Both
// of these are functions, which allow the strategy to either start a
// reconnection attempt, or to abort reconnection (which will close
 the session)
var reconnectionStrategy = function(start, abort) {
    if (attempts > maximumAttempts) {
        abort();
    } else {
        var wait = Math.min(Math.pow(2, attempts++) * 100,
 maximumAttemptInterval);

        // Wait the specified time period, and then start the
 reconnection attempt
        setTimeout(start, wait);
    }
};

// Connect to the server.
diffusion.connect({
    host : 'diffusion.example.com',
    port : 443,
    secure : true,
    principal : 'control',
    credentials : 'password',
    reconnect : {
        timeout : maximumTimeoutDuration,
        strategy : reconnectionStrategy
    }
}).then(function(session) {

    session.on('disconnect', function() {
        // This will be called when we lose connection. Because we've
 specified the
        // reconnection strategy, it will be called automatically
 when this event
```

```
        // is dispatched
    });

    session.on('reconnect', function() {
        // If the session is able to reconnect within the reconnect
 timeout, this
        // event will be dispatched to notify that normal operations
 may resume
        attempts = 0;
    });

    session.on('close', function() {
        // If the session is closed normally, or the session is
 unable to reconnect,
        // this event will be dispatched to notify that the session
 is no longer
        // operational.
    });
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */
using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Session.Reconnection;
using static System.Console;

namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Implementation that demonstrates session reconnection
 strategy.
    /// </summary>
    public sealed class SessionReconnection
    {
        private int maximumTimeoutDuration = 1000 * 60 * 10;

        private ReconnectionStrategy reconnectionStrategy = new
 ReconnectionStrategy();

        public SessionReconnection(string serverUrl)
```

```
        {
            var factory = Diffusion.Sessions;

            var session = Connect(serverUrl, factory);

            if (session != null)
            {
                WriteLine("The session has been created.");
            }

            Thread.Sleep(60000);

            session.Close();
        }

        public ISession Connect(string url, ISessionFactory
initialFactory)
        {
            try
            {
                string principal = "control";
                string password = "password";

                var factory = initialFactory
                    .Principal(principal)

.Credentials(Diffusion.Credentials.Password(password))
                    .CertificateValidation((cert, chain, errors)
                        => CertificateValidationResult.ACCEPT)
                    .ReconnectionTimeout(maximumTimeoutDuration)
                    .ReconnectionStrategy(reconnectionStrategy)

.SessionStateChangedHandler(OnSessionStateChanged);

                return factory.Open(url);
            }
            catch (Exception ex)
            {
                WriteLine($"Session connection error : {ex}.");
            }

            return null;
        }

        private void OnSessionStateChanged(object sender,
SessionListenerEventArgs e)
        {
            if (e.NewState == SessionState.RECOVERING_RECONNECT)
            {
                // The session has been disconnected, and has entered
                // recovery state. It is during this state that
                // the reconnect strategy will be called
                WriteLine("The session has been disconnected.");
            }

            if (e.NewState == SessionState.CONNECTED_ACTIVE)
            {
                // The session has connected for the first time, or
it has
                // been reconnected.
                reconnectionStrategy.Retries = 0;

                WriteLine("The session has connected.");
```

```
            }

            if (e.OldState == SessionState.RECOVERING_RECONNECT)
            {
                // The session has left recovery state. It may either
 be
                // attempting to reconnect, or the attempt has been
 aborted;
                // this will be reflected in the newState.
            }

            if (e.NewState == SessionState.CLOSED_BY_CLIENT)
            {
                WriteLine("The session has been closed.");
            }
        }

        /// <summary>
        /// A reconnection strategy that gets applied after the
 connection failure notification.
        /// </summary>
        private class ReconnectionStrategy : IReconnectionStrategy
        {
            public int Retries { get; set; }

            // Set the maximum interval between reconnect attempts to
 60 seconds.
            private long maximumAttemptInterval = 1000 * 60;

            public ReconnectionStrategy() => Retries = 0;

            public async Task
 PerformReconnection(IReconnectionAttempt reconnection)
            {
                long wait = Math.Min((long)Math.Pow(2, Retries++) *
 100L, maximumAttemptInterval);

                Thread.Sleep((int)wait);

                WriteLine("Attempting to reconnect...");

                reconnection.Start();
            }
        }
    }
}
```

Java and Android

```
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.session.Session;
import com.pushtechnology.diffusion.client.session.Session.Listener;
import com.pushtechnology.diffusion.client.session.Session.State;
import
 com.pushtechnology.diffusion.client.session.reconnect.ReconnectionStrategy;
```

```
/**
 * This example class demonstrates the ability to set a custom {@link
 ReconnectionStrategy}
 * when creating sessions.
 *
 * @author Push Technology Limited
 * @since 5.5
 */
public class ClientWithReconnectionStrategy {

    private volatile int retries = 0;
    /**
     * Constructor.
     */
    public ClientWithReconnectionStrategy() {

        // Set the maximum amount of time we'll try and reconnect for
 to 10 minutes.
        final int maximumTimeoutDuration = 1000 * 60 * 10;

        // Set the maximum interval between reconnect attempts to 60
 seconds.
        final long maximumAttemptInterval = 1000 * 60;

        // Create a new reconnection strategy that applies an
 exponential backoff
        final ReconnectionStrategy reconnectionStrategy = new
 ReconnectionStrategy() {
            private final ScheduledExecutorService scheduler =
 Executors.newScheduledThreadPool(1);

            @Override
            public void performReconnection(final ReconnectionAttempt
 reconnection) {
                final long exponentialWaitTime =
                    Math.min((long) Math.pow(2,  retries++) * 100L,
 maximumAttemptInterval);

                scheduler.schedule(new Runnable() {
                    @Override
                    public void run() {
                        reconnection.start();
                    }
                }, exponentialWaitTime, TimeUnit.MILLISECONDS);
            }
        };

        final Session session =
 Diffusion.sessions().reconnectionTimeout(maximumTimeoutDuration)

 .reconnectionStrategy(reconnectionStrategy)
                                                        .open("ws://
diffusion.example.com:80");
        session.addListener(new Listener() {
            @Override
            public void onSessionStateChanged(Session session, State
 oldState, State newState) {

                if (newState == State.RECOVERING_RECONNECT) {
                    // The session has been disconnected, and has
 entered recovery state. It is during this state that
                    // the reconnect strategy will be called
                }
```

```
                if (newState == State.CONNECTED_ACTIVE) {
                    // The session has connected for the first time,
 or it has been reconnected.
                    retries = 0;
                }

                if (oldState == State.RECOVERING_RECONNECT) {
                    // The session has left recovery state. It may
 either be attempting to reconnect, or the attempt has
                    // been aborted; this will be reflected in the
 newState.
                }
            }
        });
    }
}
```

C

```
/*
 * This example shows how to make a synchronous connection to
 * Diffusion, with user-provided reconnection logic.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#include <apr_time.h>

#include "diffusion.h"
#include "args.h"

ARG_OPTS_T arg_opts[] = {
        ARG_OPTS_HELP,
        {'u', "url", "Diffusion server URL", ARG_OPTIONAL,
 ARG_HAS_VALUE, "ws://localhost:8080"},
        {'p', "principal", "Principal (username) for the connection",
 ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'c', "credentials", "Credentials (password) for the
 connection", ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'s', "sleep", "Time to sleep before disconnecting (in
 seconds).", ARG_OPTIONAL, ARG_HAS_VALUE, "5" },
        END_OF_ARG_OPTS
};

/*
 * This callback is used when the session state changes, e.g. when a
 session
 * moves from a "connecting" to a "connected" state, or from
 "connected" to
 * "closed".
 */
static void
on_session_state_changed(SESSION_T *session,
        const SESSION_STATE_T old_state,
        const SESSION_STATE_T new_state)
{
        printf("Session state changed from %s (%d) to %s (%d)\n",
```

```
                session_state_as_string(old_state), old_state,
                session_state_as_string(new_state), new_state);
}

typedef struct {
        long current_wait;
        long max_wait;
} BACKOFF_STRATEGY_ARGS_T;

static RECONNECTION_ATTEMPT_ACTION_T
backoff_reconnection_strategy(SESSION_T *session, void *args)
{
        BACKOFF_STRATEGY_ARGS_T *backoff_args = args;

        printf("Waiting for %ld ms\n", backoff_args->current_wait);

        apr_sleep(backoff_args->current_wait * 1000); // µs -> ms

        // But only up to some maximum time.
        if(backoff_args->current_wait > backoff_args->max_wait) {
                backoff_args->current_wait = backoff_args->max_wait;
        }

        return RECONNECTION_ATTEMPT_ACTION_START;
}

static void
backoff_success(SESSION_T *session, void *args)
{
        printf("Reconnection successful\n");

        BACKOFF_STRATEGY_ARGS_T *backoff_args = args;
        backoff_args->current_wait = 0; // Reset wait.
}

static void
backoff_failure(SESSION_T *session, void *args)
{
        printf("Reconnection failed (%s)\n",
 session_state_as_string(session->state));

        BACKOFF_STRATEGY_ARGS_T *backoff_args = args;

        // Exponential backoff.
        if(backoff_args->current_wait == 0) {
                backoff_args->current_wait = 1;
        }
        else {
                backoff_args->current_wait *= 2;
        }
}

/*
 * Entry point for the example.
 */
int
main(int argc, char **argv)
{
        /*
         * Standard command-line parsing.
         */
        HASH_T *options = parse_cmdline(argc, argv, arg_opts);
        if(options == NULL || hash_get(options, "help") != NULL) {
```

```
                show_usage(argc, argv, arg_opts);
                return EXIT_FAILURE;
        }

        const char *url = hash_get(options, "url");
        const char *principal = hash_get(options, "principal");
        CREDENTIALS_T *credentials = NULL;
        const char *password = hash_get(options, "credentials");
        if(password != NULL) {
                credentials = credentials_create_password(password);
        }

        const unsigned int sleep_time = atol(hash_get(options,
"sleep"));

        SESSION_T *session;
        DIFFUSION_ERROR_T error = { 0 };

        SESSION_LISTENER_T session_listener = { 0 };
        session_listener.on_state_changed =
&on_session_state_changed;

        /*
         * Set the arguments to our exponential backoff strategy.
         */
        BACKOFF_STRATEGY_ARGS_T *backoff_args = calloc(1,
sizeof(BACKOFF_STRATEGY_ARGS_T));
        backoff_args->current_wait = 0;
        backoff_args->max_wait = 5000;

        /*
         * Create the backoff strategy.
         */
        RECONNECTION_STRATEGY_T *reconnection_strategy =

make_reconnection_strategy_user_function(backoff_reconnection_strategy,

backoff_args,

backoff_success,

backoff_failure,
                                                        NULL);

        /*
         * Only ever retry for 30 seconds.
         */
        reconnection_strategy_set_timeout(reconnection_strategy, 30 *
1000);

        /*
         * Create a session, synchronously.
         */
        session = session_create(url, principal, credentials,
&session_listener, reconnection_strategy, &error);
        if(session != NULL) {
                char *sid_str = session_id_to_string(session->id);
                printf("Session created (state=%d, id=%s)\n",
session_state_get(session), sid_str);
                free(sid_str);
        }
        else {
```

```
                printf("Failed to create session: %s\n",
 error.message);
                free(error.message);
        }

        // With the exception of backoff_args, the reconnection
 strategy is
        // copied withing session_create() and may be freed now.
        free(reconnection_strategy);

        /*
         * Sleep for a while.
         */
        sleep(sleep_time);

        /*
         * Close the session, and release resources and memory.
         */
        session_close(session, NULL);
        session_free(session);

        free(backoff_args);

        credentials_free(credentials);
        hash_free(options, NULL, free);

        return EXIT_SUCCESS;
}
```

Apple

```
@import Diffusion;

@interface ExponentialBackoffReconnectionStrategy : NSObject
 <PTDiffusionSessionReconnectionStrategy>
@end

@implementation CustomReconnectionStrategyExample {
    PTDiffusionSession* _session;
}

-(void)startWithURL:(NSURL*)url {
    NSLog(@"Connecting...");

    PTDiffusionMutableSessionConfiguration *const
 sessionConfiguration =
        [PTDiffusionMutableSessionConfiguration new];

    // Set the maximum amount of time we'll try and reconnect for to
 10 minutes.
    sessionConfiguration.reconnectionTimeout = @(10.0 * 60.0); //
 seconds

    // Set the reconnection strategy to be used.
    sessionConfiguration.reconnectionStrategy =
 [ExponentialBackoffReconnectionStrategy new];

    // Start connecting asynchronously.
    [PTDiffusionSession openWithURL:url
                      configuration:sessionConfiguration
```

```
                    completionHandler:^(PTDiffusionSession *session,
 NSError *error)
    {
        if (!session) {
            NSLog(@"Failed to open session: %@", error);
            return;
        }

        // At this point we now have a connected session.
        NSLog(@"Connected.");

        // Set ivar to maintain a strong reference to the session.
        _session = session;
    }];
}

@end

@implementation ExponentialBackoffReconnectionStrategy {
    NSUInteger _attemptCount;
}

-(void)         diffusionSession:(PTDiffusionSession *const)session
    wishesToReconnectWithAttempt:
(PTDiffusionSessionReconnectionAttempt *const)attempt {
    // Limit the maximum time to delay between reconnection attempts
 to 60 seconds.
    const NSTimeInterval maximumAttemptInterval = 60.0;

    // Compute delay for exponential backoff based on the number of
 attempts so far.
    const NSTimeInterval delay = MIN(pow(2.0, _attemptCount++) * 0.1,
 maximumAttemptInterval);

    // Schedule asynchronous execution.
    NSLog(@"Reconnection attempt scheduled for %.2fs", delay);
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(delay *
 NSEC_PER_SEC)),
        dispatch_get_main_queue(), ^
    {
        NSLog(@"Attempting reconnection.");
        [attempt start];
    });
}

@end
```

Related concepts

Session reconnection on page 505
You can configure the session reconnection feature by configuring the connectors at the Diffusion server to keep the client session in a disconnected state for a period before closing the session.

Session failover

Session failover occurs when a client that disconnects from a Diffusion server attempts to connect to a different Diffusion server that also has information about that client's session.

For session failover to occur, session replication must be configured for a cluster of Diffusion servers. For more information, see Configuring replication on page 466.

Differences between session reconnection and session failover

When a client loses a load-balanced connection to Diffusion, one of the following things can occur when the client attempts to reconnect through the load balancer:

Session reconnection

The load balancer forwards the client connection to the Diffusion server it was previously connected to, if that server is still available. For more information, see Reconnect to the Diffusion server on page 203.

Session failover

The load balancer forwards the client connection to a different Diffusion server that shares information about the client's session, if session replication is enabled between the servers.

Prefer session reconnection to session failover wherever possible by ensuring that the load balancer is configured to route all connections from a specific client to the same server if that server is available.

Session reconnection is more efficient as less data must be sent to the client and has less risk of data loss, as sent messages can be recovered, in-flight requests are not lost, and handlers do not need to be registered again.

For more information, see Routing strategies at your load balancer on page 626.

To a client the process of disconnection and subsequent reconnection has the following differences for session reconnection or session failover.

Session reconnection	Session failover
The client connects to the same Diffusion server it was previously connected to.	The client connects to a Diffusion server different to the one it was previously connected to.
The client sends its last session token to the server.	
The server authenticates the client connection or validates its session token.	
The server uses the session token to resynchronize the streams of messages between the server and client by resending any messages that were lost in transmission from a buffer of sent messages. If lost messages cannot be recovered because they are no longer present in a buffer, the server aborts the reconnection.	The server uses the session token to retrieve the session state and topic selections from the datagrid.
The server sends any messages that have been queued since the session disconnected.	The server uses the state to recover the session, uses the topic selections to match the subscribed topics, and sends the session the current topic value for each subscribed topic. Any in-flight requests made by the client session to the previous server are cancelled and the client session is notified by a callback. All handlers, including authentication handlers and update sources, that the client session had registered with the previous server are closed and receive a callback to notify them of the closure.

Pinging the Diffusion server

Ping the Diffusion server from your client. If the ping is successful it reports the round-trip time between your client and the Diffusion server.

The Diffusion client libraries and the Diffusion server include capabilities that automatically check whether the connection is active. However, there might be times when you want to check the connection from within your client code. For example, if the client is aware that the device it is hosted on has recently changed from a 3G connection to a WiFi connection.

Use the pings capability to asynchronously ping the Diffusion server.

JavaScript

```
session.pingServer().then(function(pingResult) {
    // Take action based on ping details.
});
```

.NET

```
IPings pings = session.Ping;
pings.PingServer( context, callback );
```

Java and Android

```
Pings pings = session.feature(Pings.class);
pings.pingServer(context, callback);
```

C

```
PING_USER_PARAMS_T params = {
        .on_ping_response = on_ping_response_user
};
ping_user(session, params);
```

Apple

```
[session.pings pingServerWithCompletionHandler:
    ^(PTDiffusionPingDetails *details, NSError *error)
{
    // Check error is `nil`, indicating success.
}];
```

Change the security principal and credentials associated with your client session

A client session can change the credentials it uses to authenticate with the Diffusion server at any time.

JavaScript

```
session.security.changePrincipal('principal',
 'password').then(function() {
        console.log('Authenticated as admin');
    });
```

.NET

```
security = session.Security;
security.ChangePrincipal( principal,
 Diffusion.Credentials.Password( password ), callback );
```

Java and Android

```
security = session.feature(Security.class);
security.changePrincipal(
    principal,
    Diffusion.credentials().password(password),
    callback);
```

C

```
// Specify callbacks for the change_principal request.
CHANGE_PRINCIPAL_PARAMS_T params = {
        .principal = hash_get(options, "principal"),
        .credentials = credentials,
        .on_change_principal = on_change_principal,
        .on_change_principal_failure = on_change_principal_failure
};

// Do the change.
change_principal(session, params);
```

Apple

```
// Create a credentials object encapsulating a string password.
PTDiffusionCredentials *const credentials =
    [[PTDiffusionCredentials alloc] initWithPassword:@"password"];

// Use the Security feature from your session...
[session.security changePrincipal:@"principal"
                      credentials:credentials
                completionHandler:^(NSError *error)
{
    // Check error is `nil`, indicating success.
```

```
}];
```

When the principal associated with a session changes, the following happens:

- The $Principal session property is updated to contain the new principal.
- The roles associated with the old principal are removed from the session and those roles associated with the new principal are assigned to the session.
- Topic subscriptions made with the old principal are not re-evaluated. The session remains subscribed to any topics the new principal does not have permissions for.

Session properties

A client session has a number of properties associated with it. Properties are key-value pairs. Both the key and the value are case sensitive.

Session properties provide a powerful way for client sessions to target actions at a specific session or set of sessions whose session properties match a given criteria. Client sessions can use session filtering to select a set of client sessions upon which to perform one of the following actions:

- Send messages directly to that session or set of sessions.

  For more information, see Sending request messages to a session filter on page 338.
- Subscribe that session or set of sessions to a topic.

  For more information, see Managing subscriptions on page 309.
- Unsubscribe that session or set of sessions from a topic.

  For more information, see Managing subscriptions on page 309.

For more information, see Session filtering on page 222.

A client session with the appropriate permissions can also view all of the session properties and modify the user-defined session properties. For more information, see Working with session properties on page 378.

Fixed properties

Fixed properties are set by the Diffusion server when a client opens a session with it. Fixed property keys are prefixed by a dollar sign ($). The fixed session properties are:

$SessionId

The session identifier.

$Principal

The security principal the session uses to connect to the Diffusion server.

$Roles

Authorization roles assigned to the session, represented as quoted strings (for example "client", "topic_control"). For more information, see Role-based authorization on page 138.

$ClientType

The client type of the session. For more information, see Client types on page 123.

$Transport

The transport the client session uses to connect to the Diffusion server. For more information, see Client types on page 123.

$ServerName

The name of the Diffusion server that the client connects to.

$Connector

The name of the connector on which the client connected to the Diffusion server.

$Country

The two letter country code for the country where the client's internet address is located. The value is uppercase.

$Language

The two letter language code for the most common language of the country where the client's internet address is located. The value is lowercase.

$ClientIP

The session's IP address represented as a string.

$Latitude

The client session's geographic latitude, if this can be ascertained.

$Longitude

The client session's geographic longitude, if this can be ascertained.

$MQTTClientId

The MQTT client identifier. Only set for MQTT sessions. If present, the value of the $ClientType session property will be MQTT.

$StartTime

The client session's start time in milliseconds since the epoch.

User-defined properties

An authentication handler that allows the client session to connect can assign additional properties to the session. The keys of these properties are case sensitive, non-empty strings, and cannot contain any of ' ', '\t', '\r', '\n', '"', ''', '(', ')'

Client-proposed properties

A client can propose user-defined session properties when it opens a session. You can configure the system authentication handler to accept client-proposed properties. You can also write an authentication handler with the `Authenticator` interface to accept client-proposed properties and apply them to the session.

Related concepts

Session filtering on page 222
Session filters enable you to query the set of connected client sessions on the Diffusion server based on their session properties.

Managing subscriptions on page 309
A client can use the SubscriptionControl feature to subscribe other client sessions to topics that they have not requested subscription to themselves and also to unsubscribe clients from topics. It also enables the client to register as the handler for routing topic subscriptions.

Managing sessions on page 377

A client session with the appropriate permissions can receive notifications and information about other client sessions. A client session with the appropriate permissions can also manage other client sessions.

Session filtering

Session filters enable you to query the set of connected client sessions on the Diffusion server based on their session properties.

To perform an action on a subset of the connected client sessions, you can create a query expression that filters the set of connected client sessions by the values of their session properties. Filter query expressions are parsed and evaluated by the Diffusion server.

The query expression used to filter the session is made up of one or more clauses chained together by boolean operators.

Creating a single search clause

Search clauses have the following form:

```
key operator 'value'
```

key

The key name of the session property to be tested. The key name is case sensitive. Not required with the HASROLES operator.

operator

The operator that defines the test to be performed. The operator is not case sensitive.

value

The test value to be compared to the session property value. This value is a string and must be contained in single or double quotation marks. Any special characters must be escaped with Java escaping. The value is case sensitive.

Table 30: Session filter search clause operators

Operator	Description
IS	Tests whether the session property value associated with the property key matches the test value.
EQ	Equals. Tests whether the session property value associated with the property key matches the test value. Equivalent to 'IS'.
NE	Not equal. Tests whether the session property value associated with the key is not equal to the test value.
IN	Tests whether the value of a session property belongs to a set, which you provide after the operator.
HASROLES	Tests whether a session has any one of a set of security roles, which you provide after the operator.

You can use the special 'all' clause to match all sessions. This does not take a key or a value so it is always simply:

```
all
```

The 'all' clause can be useful when creating a session metric collector.

Examples: single search clause

Filter by clients that connect with the principal Ellington:

```
$Principal IS 'Ellington'
```

Filter by clients that connect to the Diffusion server using WebSocket:

```
$Transport EQ 'WEBSOCKET'
```

Filter by clients that are not located in the United Kingdom:

```
$Country NE 'GB'
```

Filter by clients that have the user-defined property Location set to San Jose:

```
Location IS "San Jose"
```

Filter by clients that have the user-defined property Status set to Active:

```
Status EQ 'Active'
```

Filter by clients that do not have the user-defined property Tier set to Premium:

```
Tier NE 'Premium'
```

Filter by clients that are located in one of a set of countries:

```
$COUNTRY in ['UK', 'DE', 'FR']
```

Filter by clients that have either of the specified security roles:

```
hasRoles ["operator", "trading desk"]
```

Chaining multiple clauses

Chain individual clauses together using boolean operator or use the NOT operator to negate a search clause. Boolean operators are not case sensitive.

Table 31: Session filter boolean operators

Operator	Description
AND	Specifies that both joined search clauses must be true.
OR	Specifies that at least one of the joined search clauses must be true.
NOT	Specifies that the following search clause or set of search clauses must not be true.

Use parentheses to group sets of clauses and indicate the order of precedence for evaluation. If no order of precedence is explicitly defined, the AND operator takes precedence over the OR operator.

Examples: multiple search clauses

Filter by clients that connect with one of the principals Fitzgerald, Gillespie, or Hancock:

```
$Principal IS 'Fitzgerald' OR $Principal IS 'Gillespie' OR $Principal
 IS 'Hancock'
```

Filter by clients that connect to the Diffusion server using WebSocket and are located in France and have the user-defined property Status set to Active:

```
$Transport EQ 'WEBSOCKET' AND $Country IS 'FR' AND Status EQ 'Active'
```

Filter by clients that are located in the United States, but do not connect with either of the principals Monk or Peterson:

```
$Country EQ 'US' AND NOT ($Principal IS 'Monk' OR $Principal IS
 'Peterson')
```

Filter by clients excluding those that have both the user-defined property Status set to Inactive and the user-defined property Tier set to Free:

```
NOT (Status IS 'Inactive' AND Tier IS 'Free')
```

Related concepts

Session properties on page 220
A client session has a number of properties associated with it. Properties are key-value pairs. Both the key and the value are case sensitive.

Managing subscriptions on page 309
A client can use the SubscriptionControl feature to subscribe other client sessions to topics that they have not requested subscription to themselves and also to unsubscribe clients from topics. It also enables the client to register as the handler for routing topic subscriptions.

Managing sessions on page 377
A client session with the appropriate permissions can receive notifications and information about other client sessions. A client session with the appropriate permissions can also manage other client sessions.

Receiving data from topics

A client can subscribe to a topic to receive a stream of updates or can fetch the current state of a topic.

Topics

A topic is a logical channel through which data can be distributed to clients. Topics provide a logical link between publishing clients and subscribing clients.

Diffusion provides different types of topic that can be used to stream diffent data formats or can be used for special purposes. For more information about topic types and uses, see Topics on page 69.

Subscribing

To receive data published to a topic as a stream of updates, a client takes the following actions:

- Subscribe to a topic or set of topics.
- Register a stream that matches the topic or set of topics, or register a matching fallback stream.

For topics that exist and that the client is subscribed to, data is received through a stream that is registered against that topic - if one exists and is of the appropriate type.

> **Note:** The subscription flow is fully described in the design section of the documentation. For more information, see Subscribing to topics on page 96.

Fetching

To fetch the current value of a topic or set of topics, a client makes a fetch request, passing in a fetch stream, and uses the fetch stream to receive the data.

Streams

Clients use streams to receive data from topics.

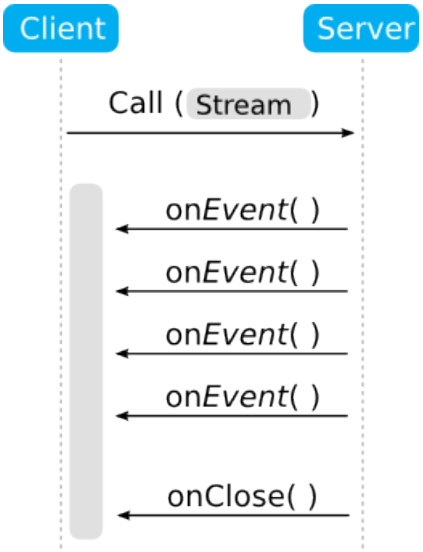


Figure 19: A stream

Streams are API objects that can receive multiple calls from the Diffusion server on the same method while the stream is open. After the stream is closed or discarded, it can no longer receive responses on any of its other methods.

Subscribing to topics

Subscribe to topics with topic selectors. When topics exist that match the selections made by the client, data from those topics is sent to the client from the Diffusion server.

The client must register a stream to access topic data that has been sent from the Diffusion server. For more information, see Subscribing to topics on page 96.

Required permissions: select_topic and read_topic permissions for the specified topics

Subscribing to topics

A client can subscribe to a topic to receive updates that are published to the topic. If the topic has state, when the client subscribes to that topic the Diffusion server sends the topic state as a full value. Subsequent updates to the data on the topic are sent as deltas or as full values depending on the type of the topic and the structure of its data.

JavaScript

```
session.select('topic_selector');
```

.NET

```
topics.Subscribe( topic, new TopicsCompletionCallbackDefault() );
```

Java and Android

```
topics.subscribe(topic_selector).whenComplete((voidResult, exception)
 -> {
    //Do something
});
```

C

```
// Define the required callbacks elsewhere
subscribe(session, (SUBSCRIPTION_PARAMS_T) { .topic_selector =
 topic_selector,
                                             .on_topic_message =
 on_topic_message,
                                             .on_subscribe =
 on_subscribe });
```

Apple

```
[session.topics subscribeWithTopicSelectorExpression:topic_selector
                                    completionHandler:^(NSError *
 const error)
{
    if (error) {
        NSLog(@"Subscribe request failed. Error: %@", error);
    } else {
        NSLog(@"Subscribe request succeeded.");
    }
}];
```

A client can subscribe to multiple topics in a single request by using topic selectors. Topic selectors enable you to select whole branches of the topic tree or use regular expressions to select topics based on the names in the topic path.

For more information, see Topic selectors on page 44.

Unsubscribing from topics

To stop receiving updates from a topic or set of topics, unsubscribe from the topic or topics:

JavaScript

```
session.unsubscribe('topic_selector');
```

.NET

```
topics.Unsubscribe( topic, new TopicsCompletionCallbackDefault() );
```

Java and Android

```
topics.unsubscribe(topic_selector).whenComplete((voidResult,
 exception) -> {
    //Do something
});
```

C

```
// Define an on_unsubscribe callback elsewhere
unsubscribe(session, (UNSUBSCRIPTION_PARAMS_T) {.topic_selector =
 topic_selector,
                                                 .on_unsubscribe =
 on_unsubscribe} );
```

Apple

```
[session.topics unsubscribeFromTopicSelectorExpression:topic_selector
                                      completionHandler:^(NSError *
 const error)
    {
        if (error) {
            NSLog(@"Unsubscribe request failed. Error: %@", error);
        } else {
            NSLog(@"Unsubscribe request succeeded.");
        }
    }];
```

Using streams for subscription

Register a stream against a set of topics to access values published to those topics. For a registered stream to access the value of a topic, the topic type must match the stream and the client must be subscribed to the topic.

Subscribing to a topic causes the value of the topic to be sent from the Diffusion server to the client. Registering a stream that matches the topic enables the client to access these values. For more information, see Subscribing to topics on page 96

Two kinds of stream are provided to receive updates from subscribed topics: value streams and topic streams.

Value streams

Value streams are typed. Register value streams against a set of topics by using a topic selector. A value stream receives updates for any subscribed topics that match the value stream's type and the topic selector used when registering the value stream.

A value stream can have one of the following types:

JSON

JSON topics are routed to this type of stream.

Binary

Binary topics are routed to this type of stream.

String

String topics are routed to this type of stream.

Int64

Int64 topics are routed to this type of stream.

Double

Double topics are routed to this type of stream.

RecordV2

RecordV2 topics are routed to this type of stream.

Content

JSON, binary, string, int64, double, recordV2 and single value topics are routed to this type of stream.

If a value stream receives a delta update, this delta is automatically applied to a locally cached value so that the stream always receives full values.

Using a value stream

Register the typed stream against the topic or topics that you want the stream to receive updates from:

JavaScript

```
// Register a JSON value stream
session.addStream('topic_selector', diffusion.datatypes.json())
       .on('value', function(path, specification, newValue, oldValue)
 {
            // Action to take when update is received
       });

// Register a binary value stream
session.addStream('topic_selector', diffusion.datatypes.binary())
       .on('value', function(path, specification, newValue, oldValue)
 {
            // Action to take when update is received
       });
```

.NET

```
var topics = session.Topics;

// Register a JSON value stream
topics.AddStream( "topic_selector", new
 Topics.DefaultValueStream<IJSON>() );

// Register a binary value stream
topics.AddStream( "topic_selector", new
 Topics.DefaultValueStream<IBinary>() );
```

Java and Android

```
final Topics topics = session.feature(Topics.class);

// Register a JSON value stream
topics.addStream(topic_selector, JSON.class, new
 Topics.ValueStream.Default<JSON>());

// Register a binary value stream
topics.addStream(topic_selector, Binary.class, new
 Topics.ValueStream.Default<Binary>());
```

Apple

```
// Register a JSON value stream
PTDiffusionValueStream *const jsonValueStream = [PTDiffusionJSON
 valueStreamWithDelegate:self];
    [session.topics addStream : jsonValueStream,
                 withSelector : topic_selector];

// Register a binary value stream
PTDiffusionValueStream *const binaryValueStream = [PTDiffusionBinary
 valueStreamWithDelegate:self];
    [session.topics addStream : binaryValueStream,
                 withSelector : topic_selector];
```

Use topic selectors to register the stream against multiple topics. For more information, see Topic selectors on page 44.

The examples above show how to register a default or no-op value stream against a set of topics. The stream receives values from any topic in the set whose topic data type matches the stream data type.

To make use of the values sent to your client, implement a value stream that takes the required action when an update is received from a subscribed topic that matches the type of the stream:

JavaScript

```
session.addStream('topic_selector', diffusion.datatypes.json())
    .on({
    value : function(topic, specification, newValue, oldValue) {
        console.log('Update from: ', topic, newValue.get());
    },
    subscribe : function(topic, specification) {
        console.log('Subscribed to: ', topic);
    },
    unsubscribe : function(topic, specification, reason) {
```

```
        console.log('Unsubscribed from: ', topic);
    }});
```

.NET

```
/// Basic implementation of the IValueStream for JSON topics.

private sealed class JSONStream : IValueStream<IJSON> {

/// Notification of stream being closed normally.

public void OnClose()
=> WriteLine( "The subscrption stream is now closed." );

/// Notification of a contextual error related to this callback.

/// Situations in which OnError is called include the session being
 closed, a communication
/// timeout, or a problem with the provided parameters. No further
 calls will be made to this callback.

public void OnError( ErrorReason errorReason )
=> WriteLine( $"An error has occured : {errorReason}." );

/// Notification of a successful subscription.

public void OnSubscription( string topicPath, ITopicSpecification
 specification )
=> WriteLine( $"Client subscribed to {topicPath}." );

/// Notification of a successful unsubscription.

public void OnUnsubscription( string topicPath, ITopicSpecification
 specification, TopicUnsubscribeReason reason )
=> WriteLine( $"Client unsubscribed from {topicPath} : {reason}." );

/// Topic update received.

public void OnValue( string topicPath, ITopicSpecification
 specification, IJSON oldValue, IJSON newValue )
=> WriteLine( $"New value of {topicPath} is
 {newValue.ToJSONString()}." );
}
```

Java and Android

```
private class JSONStream extends ValueStream.Default<JSON> {
    @Override
    public void onValue(
        String topicPath,
        TopicSpecification specification,
        JSON oldValue,
        JSON newValue) {
        LOG.info(newValue.toJsonString());
    }
}
```

Apple

```
@implementation JSONSubscribeExample
 (PTDiffusionJSONValueStreamDelegate)

-(void)     diffusionStream:(PTDiffusionStream *const)stream
    didSubscribeToTopicPath:(NSString *const)topicPath
              specification:(PTDiffusionTopicSpecification
 *const)specification {
    NSLog(@"Subscribed: %@", topicPath);
}

-(void)diffusionStream:(PTDiffusionValueStream *const)stream
    didUpdateTopicPath:(NSString *const)topicPath
         specification:(PTDiffusionTopicSpecification
 *const)specification
               oldJSON:(PTDiffusionJSON *const)oldJSON
               newJSON:(PTDiffusionJSON *const)newJSON {

    NSError * error;
    NSDictionary *const map = [newJSON objectWithError:&error];
    if (!map) {
        NSLog(@"Failed to create map from received JSON. Error: %@",
 error);
        return;
    }

    // For the purposes of a meaningful example, only emit a log line
 if we
    // have a rate for GBP to USD.
    if ([currency isEqualToString:@"GBP"]) {
        const id rate = map[@"USD"];
        if (rate) {
            NSLog(@"Rate for GBP to USD: %@", rate);
        }
    }
}

-(void)         diffusionStream:(PTDiffusionStream *const)stream
    didUnsubscribeFromTopicPath:(NSString *const)topicPath
                  specification:(PTDiffusionTopicSpecification
 *const)specification
                         reason:(const
 PTDiffusionTopicUnsubscriptionReason)reason {
    NSLog(@"Unsubscribed: %@", topicPath);
}

@end
```

Topic streams

> **Note:** Where a value stream is available for your topic type, we recommend you use a value stream instead of a topic stream.

Topic streams are not typed and are used to receive value and delta updates for all subscribed topics that match the topic selectors used when registering the value stream.

This type of stream provides the value and the deltas but relies upon the application to apply the deltas to a client-maintained current value. It is important, when using a topic stream with a record topic, to register the stream before subscribing to the topic. This ensures that a full value is received by the subscribing client.

Using a topic stream

Register the stream against the topic or topics that you want the stream to receive updates from:

JavaScript

```
session.stream('topic_selector')
       .on('update', function(update, topic) {
            // Do something
       });
```

.NET

```
var topics = session.Topics;

// Add a topic stream that you implemented elsewhere
topics.AddTopicStream( topic_selector, myTopicStream );
```

Java and Android

```
Topics topics = session.feature(Topics.class);
// Add a topic stream that you implemented elsewhere
topics.addTopicStream(topic_selector, new myTopicStream(data));
```

Apple

```
// Register self as the handler for topic updates on a set of topics.
[session.topics addTopicStreamWithSelector : topic_selector,
                                    delegate : self];
```

Use topic selectors to register the stream against multiple topics. For more information, see Topic selectors on page 44.

Registering a fallback stream

You can register one or more fallback streams to receive updates to subscribed topics that do not have a value stream or topic stream registered against them:

JavaScript

```
session.addFallbackStream(diffusion.datatypes.json())
     .on('value', function(topic, specification, newValue, oldValue)
 {
         // Do something
     });
```

.NET

```
var topics = session.Topics;

topics.AddFallbackStream<IJSON>( new
 Topics.DefaultValueStream<IJSON>() );
```

Java and Android

```
final Topics topics = session.feature(Topics.class);

topics.addFallbackStream(topic_selector, JSON.class, new
 Topics.ValueStream.Default());
```

C

```
/*
 * Install a global topic handler to capture messages for
 * topics we haven't explicitly subscribed to, and therefore
 * don't have a specific handler for.
 */
session->global_topic_handler = on_unexpected_topic_message;
```

Apple

```
// Register self as the fallback handler for JSON value updates.
PTDiffusionValueStream *const valueStream = [PTDiffusionJSON
 valueStreamWithDelegate:self];
[session.topics addFallbackStream:valueStream];
```

A fallback value stream receives all updates for topics of the matching type that do not have a stream already registered against them.

A fallback topic stream receives all updates for topics of any type that do not have a stream already registered against them.

Fetching the current value of a topic

A client can send a fetch request for the values and/or topic specifications of a set of topics. The result set can be filtered by topic selector and topic path range.

Required permissions: select_topic and read_topic permissions for the specified topics

A fetch request enables you to retrieve values and/or topic specifications for a set of topics without subscribing to the topics.

In Diffusion 6.2, there is a new enhanced fetch API, and the old fetch API is deprecated. This documentation describes the new API.

> **Note:** The old fetch API triggered missing topic notifications. The new enhanced API does not cause missing topic notifications.

Results can be filtered by topic selector and topic path range. A request can specify the maximum number of results to return as well as a maximum topic path depth, avoiding the inefficient transfer of very large result sets.

A request can specify the value type of topics to be returned, in which case only topics of types compatible with the given value type will be returned. Returned values will be typed accordingly, avoiding the need for data conversion.

Here is how to fetch the value of a topic or set of topics by using a topic selector to make a fetch request:

.NET

```
var session = Diffusion.Sessions
    .Principal( "client" )
    .Password( "password" )
    .Open( serverUrl );
var topics = session.Topics;
var result = await topics.FetchRequest
    .WithValues<string>()
    .FetchAsync( "*.*" );
```

Java and Android

```
session =
 Diffusion.sessions().principal("client").password("password") .open(serverUrl);

    topics = session.feature(Topics.class);

    FetchResult<String> result =
        topics.fetchRequest()
            .withValues(String.class)
            .fetch("*.*").get(5, SECONDS);
```

C

```
static int on_fetch_result(const DIFFUSION_FETCH_RESULT_T
 *fetch_result, void *context)
{
    char *result;
    LIST_T *results =
 diffusion_fetch_result_get_topic_results(fetch_result);

    DIFFUSION_TOPIC_RESULT_T *topic_result =
 list_get_data_indexed(results, 0);
    DIFFUSION_VALUE_T *value =
 diffusion_topic_result_get_value(topic_result);

    read_diffusion_string_value(value, &result, NULL);
    printf("Fetch Result: %s\n", result);

    free(result);

    return HANDLER_SUCCESS;
    }
```

To get the result set and print the results:

.NET

```
foreach ( var item in result.Results ) {
    Console.WriteLine( $"{item.Type} : {item.Path}" );
}
```

Java and Android

```
List<TopicResult<Void>> results = result.results();
    results.forEach(t -> {
         System.out.println(t.type() + " : " + t.path());
    });
```

C

```
DIFFUSION_FETCH_REQUEST_T *fetch_request =
 diffusion_fetch_request_init(session);
DIFFUSION_DATATYPE dt = DATATYPE_STRING;

SET_T *topic_types = set_new_int(1);
TOPIC_TYPE_T topic_type_string = TOPIC_TYPE_STRING;

diffusion_fetch_request_topic_types(fetch_request, topic_types,
 NULL);
diffusion_fetch_request_with_values(fetch_request, &dt, NULL);
diffusion_fetch_request_from(fetch_request, "test-fetch-query",
 NULL);
diffusion_fetch_request_to(fetch_request, "test-fetch-query", NULL);
diffusion_fetch_request_first(fetch_request, 1, NULL);
diffusion_fetch_request_maximum_result_size(fetch_request, 100,
 NULL);

DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
    .topic_selector = ">test-fetch-query",
    .fetch_request = fetch_request,
    .on_fetch_result = on_fetch_result
};

diffusion_fetch_request_fetch(session, params);
```

Fetching topic specifications

You can return topic specifications instead of values for each topic selected.

.NET

```
var result = await topics.FetchRequest
    .WithProperties()
    .FetchAsync( "*Accounts/" );
var topicResult = result.Results.First();
var properties = topicResult.Specification.Properties;
```

Java and Android

```
FetchResult<Void> result =
    topics.fetchRequest()
        .withProperties()
        .fetch("*Accounts/").get(5, SECONDS);

TopicResult<Void> topicResult = result.get(0);
Map<String, String> properties =
 topicResult.specification().getProperties();
```

C

```
static int on_fetch_result(const DIFFUSION_FETCH_RESULT_T
 *fetch_result, void *context)
{
    LIST_T *results =
 diffusion_fetch_result_get_topic_results(fetch_result);

    DIFFUSION_TOPIC_RESULT_T *topic_result =
 list_get_data_indexed(results, 0);
    TOPIC_SPECIFICATION_T *spec =
 diffusion_topic_result_get_specification(topic_result);

    list_free(results, (void (*)(void
 *))diffusion_topic_result_free);
    return HANDLER_SUCCESS;
}

                        ...

DIFFUSION_FETCH_REQUEST_T *fetch_request =
 diffusion_fetch_request_init(session);
diffusion_fetch_request_with_properties(fetch_request, NULL);

DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
    .topic_selector = "*Accounts/",
    .fetch_request = fetch_request,
    .on_fetch_result = on_fetch_result
};

diffusion_fetch_request_fetch(session, params);
```

Filtering by topic type

The results can also be restricted to topics of a particular topic type or types:

.NET

```
var result = await topics.FetchRequest
    .TopicTypes( new[] { TopicType.STRING, TopicType.INT64 } )
    .FetchAsync( "*Accounts/" );
```

Java and Android

```
```

```
FetchResult<Void> result =
    topics.fetchRequest()
        .topicTypes(EnumSet.of(TopicType.STRING, TopicType.INT64))
        .fetch("*Accounts/").get(5, SECONDS);
```

C

```
static int on_fetch_result(const DIFFUSION_FETCH_RESULT_T
 *fetch_result, void *context)
{
    LIST_T *results =
 diffusion_fetch_result_get_topic_results(fetch_result);

    DIFFUSION_TOPIC_RESULT_T *topic_result =
 list_get_data_indexed(results, 0);
    TOPIC_SPECIFICATION_T *spec =
 diffusion_topic_result_get_specification(topic_result);

    list_free(results, (void (*)(void
 *))diffusion_topic_result_free);
    return HANDLER_SUCCESS;
}

...

DIFFUSION_FETCH_REQUEST_T *fetch_request =
 diffusion_fetch_request_init(session);
diffusion_fetch_request_with_properties(fetch_request, NULL);

DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
    .topic_selector = "*Accounts/",
    .fetch_request = fetch_request,
    .on_fetch_result = on_fetch_result
};

diffusion_fetch_request_fetch(session, params);
```

Restricting the results to a range of topics

You can restrict the returned results to within a specified range of topics. All the topics within the selection that have a path that is lexically within the specified range will be returned, at all levels.

You can specify either a start point or an end point or both. For example, if you specify a start point but no end point, results will be returned from the start point up to the end of the topic tree.

The specified start and end points do not need to represent topics that actually exist.

.NET

```
var result = await topics.FetchRequest
    .WithValues<IBytes>()
    .From( "Accounts/Dept05" )
    .To( "Accounts/Dept10" )
    .FetchAsync( "*Accounts/" );
```

Java and Android

```
FetchResult<Bytes> result =
    topics.fetchRequest()
        .withValues(Bytes.class)
        .from("Accounts/Dept05")
        .to("Accounts/Dept10")
        .fetch("*Accounts/").get(5, SECONDS);
```

C

```
static int on_fetch_result(const DIFFUSION_FETCH_RESULT_T
 *fetch_result, void *context)
{
    LIST_T *results =
 diffusion_fetch_result_get_topic_results(fetch_result);
    return HANDLER_SUCCESS;
}

...

DIFFUSION_FETCH_REQUEST_T *fetch_request =
 diffusion_fetch_request_init(session);

diffusion_fetch_request_with_values(fetch_request, NULL, NULL);
diffusion_fetch_request_from(fetch_request, "Accounts/Dept05", NULL);
diffusion_fetch_request_to(fetch_request, "Accounts/Dept10", NULL);

DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
    .topic_selector = "*Accounts/",
    .fetch_request = fetch_request,
    .on_fetch_result = on_fetch_result
};

diffusion_fetch_request_fetch(session, params);
```

This example will return all topics under `Accounts` from `Accounts/Dept05` to `Accounts/Dept10` inclusive.

Paging through topics

You can specify a non-inclusive range using the `after` and `before` methods. You can limit the number of results and check if there are further results remaining.

By combining these, you can page through a topic tree. This can be useful when there is a large number of topics and you wish to access it in manageable chunks, for example when presenting results from a large set into a limited window in a user interface.

Here is an example of paging through all string topics in a topic tree, in chunks of 20:

.NET

```
var request = topics.FetchRequest
    .WithValues<string>()
    .First( 20 );
var result = await request.FetchAsync( "*.*" );
if ( result.HasMore ) {
```

```
    result = await request
        .After( result.Results.Last().Path )
        .FetchAsync( "*.*" );
}
```

Java and Android

```
FetchRequest request =
    topics.fetchRequest()
        .withValues(String.class)
        .first(20);
    FetchResult<String> result = request.fetch("*.*").get(5,
 SECONDS);
    if (result.hasMore()) {
        result =
 request.after(result.results.get(19).path()).fetch("*.*");
    }
```

C

```
DIFFUSION_FETCH_REQUEST_T *fetch_request = NULL;

static int on_fetch_result(const DIFFUSION_FETCH_RESULT_T
 *fetch_result, void *context)
{
    if(diffusion_fetch_result_has_more(fetch_result)) {
        LIST_T *results =
 diffusion_fetch_result_get_topic_results(fetch_result);
        DIFFUSION_TOPIC_RESULT_T *topic_result =
 list_get_data_indexed(results, 19);
        diffusion_fetch_request_after(fetch_request,
 diffusion_topic_result_get_path(topic_result), NULL);

         DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
             .topic_selector = "*.*",
             .fetch_request = fetch_request,
             .on_fetch_result = on_fetch_result
         };

         diffusion_fetch_request_fetch(session, params);
         list_free(results, (void (*)(void
 *))diffusion_topic_result_free);
    }

    return HANDLER_SUCCESS;
}

 ...

 fetch_request = diffusion_fetch_request_init(session);
 DIFFUSION_DATATYPE dt = DATATYPE_STRING;

 diffusion_fetch_request_with_values(fetch_request, &dt, NULL);
 diffusion_fetch_request_first(fetch_request, 20, NULL);

 DIFFUSION_FETCH_REQUEST_PARAMS_T params = {
     .topic_selector = "*.*",
     .fetch_request = fetch_request,
```

```
    .on_fetch_result = on_fetch_result
};

diffusion_fetch_request_fetch(session, params);
```

Fetching unpublished delayed topics

Delayed topic views create reference topics in an unpublished state. The topics are published once the delay time has expired.

A client session can use a fetch request with the `withUnpublishedDelayedTopics` method to retrieve any unpublished topics.

A reference topic in the unpublished state which matches the query will only be included in the fetch results if the session has `read_topic` permission for the reference's source topic as well as `read_topic` permission for the reference topic.

Receiving topic notifications

Receive topic notifications using topic selectors. This enables a client to receive updates when topics are added or removed, without the topic values.

> **Note:** Topic notifications are supported by the Android API, Java API and JavaScript API.

The client must register a listener object to receive notifications about selected topics. Use a topic selector to specify the topics.

For more details about topic notifications, see Topic notifications on page 98.

Required permissions: select_topic and read_topic permissions for the specified topics

Receiving topic notifications

A client can register to receive notifications about a set of topics via a listener object.

JavaScript

```
var listener = {
    onDescendantNotification: function(topicPath, type) {},
    onTopicNotification: function(topicPath, topicSpecification,
 type) {},
    onClose: function() {},
    onError: function(error) {}
};

session.notifications.addListener(listener).then(function(reg) {
    reg.select("foo");
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
```

```
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Features.Control.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation of a listener for topic notifications.
    /// </summary>
    public sealed class TopicNotificationListener {
        private const string TOPIC_PREFIX = "topic-notifications";

        public TopicNotificationListener(string serverUrl) {
            var selector = $"?{TOPIC_PREFIX}//";

            var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open( serverUrl );

            var notifications = session.TopicNotifications;

            INotificationRegistration registration = null;
            string path = string.Empty;

            // Register a listener to receive topic notifications.
            try
            {
                registration = await
 notifications.AddListenerAsync(new Listener());
            }
            catch(Exception ex)
            {
                WriteLine($"Failed to add listener : {ex}.");
                session.Close();
                return;
            }

            // Start receiving notifications.
            try
            {
                await registration.SelectAsync(selector);
            }
```

```
            catch (Exception ex)
            {
                WriteLine($"Selector '{selector}' registration
 failed : {ex}.");
                session.Close();
                return;
            }

            // Add a topic.
            try
            {
                path = $"{TOPIC_PREFIX}/
{DateTime.Now.ToFileTimeUtc()}";
                var specification =
 session.TopicControl.NewSpecification(TopicType.STRING);

                await session.TopicControl.AddTopicAsync(path,
 specification);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add topic '{path}' : {ex}.");
                session.Close();
                return;
            }

            // Remove the topic.
            try
            {
                await session.TopicControl.RemoveTopicsAsync(path);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to remove topic '{path}' :
 {ex}.");
                session.Close();
                return;
            }

            // Stop receiving notifications.
            try
            {
                await registration.DeselectAsync(selector);
            }
            catch (Exception ex)
            {
                WriteLine($"Deselection failed for selector
 '{selector}' : {ex}.");
                session.Close();
                return;
            }

            // Unregister the listener.
            try
            {
                await registration.CloseAsync();
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to unregister the listener :
 {ex}.");
            }
```

```
            // Close the session
            session.Close();
        }

        /// <summary>
        /// The listener for topic notifications.
        /// </summary>
        private class Listener : ITopicNotificationListener {
            /// <summary>
            /// Indicates that the stream was closed.
            /// </summary>
            public void OnClose()
            {
                WriteLine("The listener was closed.");
            }

            /// <summary>
            /// Notification for an immediate descendant of a
 selected topic path.
            /// </summary>
            public void OnDescendantNotification(string topicPath,
 NotificationType type)
            {
                WriteLine($"Descendant topic '{topicPath}' has been
 {type}.");
            }

            /// <summary>
            /// Indicates an error received by the callback.
            /// </summary>
            public void OnError(ErrorReason errorReason)
            {
                WriteLine($"The listener received the error:
 '{errorReason}'.");
            }

            /// <summary>
            /// Notification for a selected topic.
            /// </summary>
            public void OnTopicNotification(string topicPath,
 ITopicSpecification specification,
                                            NotificationType type)
            {
                WriteLine($"Topic '{topicPath}' has been {type}.");
            }
        }
    }
}
```

Java and Android

```
final TopicNotifications notifications =
 session.feature(TopicNotifications.class);

final TopicNotificationListener listener = new
 TopicNotificationListener() {
    @Override
    public void onTopicNotification(String topicPath,
 TopicSpecification specification, NotificationType type) {
        // Handle notifications for selected/deselected topics
    }
```

```
    @Override
    public void onDescendantNotification(String topicPath,
 NotificationType type) {
        // Handle notifications for immediate descendants
    }


    @Override
    public void onClose() {
        // The listener has been closed
    }

    @Override
    public void onError(ErrorReason error) {
        // The listener has encountered an error
    }
};

final CompletableFuture<NotificationRegistration> future =
 notifications.addListener(listener);
final NotificationRegistration registration = future.get();

registration.select("foo");
```

Managing topics

A client can add and remove topics at the Diffusion server.

Required permissions: modify_topic

Adding topics is an asynchronous operation and calls back to notify of either successful creation of the topic or failure to create the topic.

If the topic add fails at the Diffusion server, the reason for failure is returned. Possible reasons for failure include the following:

- The topic already exists at the Diffusion server
- The name of the supplied topic is not valid
- The supplied details are not valid. This can occur only if properties are supplied.
- Permission to create the topic was denied
- An error occurred trying to initialize the newly created topic with the supplied content, possibly because it was not validly formatted

> **Note:** Sometimes you may wish to "add or update" a topic: add it if it does not exist, but update if it does. There is no single method for this. Instead, the updating client can try to add the topic first, and if it receives a response that the topic exists, can then update it.

A client can create topics subordinate to topics created by another client.

> **Note:**
>
> It is not currently possible to add new topics under branches of the topic tree that have been created by internal publishers..

Currently all topics created using a client have a lifespan the same as the Diffusion server (unless persistence is enabled). The topics remain at the Diffusion server even after the client session that created them has closed unless you explicitly specify that the topic is removed with the session.

Adding topics with topic specifications

The recommended way to add topics is to use a topic specification. Some deprecated topic types use topic details.

Adding topics with topic specifications

Required permissions: modify_topic

To create most topic types, you can either create the topic by defining just the topic type or use the more complex topic specification to specify other properties of the topic.

A topic is specified in terms of its type and a map of optional property settings which can alter the default behavior of the topic.

You can use the same instance of topic specification to create many topics.

Adding topics with topic details (deprecated)

Required permissions: modify_topic

Only use topic details if you are creating the deprecated record or single value topic types.

Clients can use full topic details to describe a topic when creating it. Builders (and convenience methods) are available for creating details relating to all the different topic types.

You can use the same instance of topic details to create many topics. This is recommended when many topics with the same definition are to be created, because caching optimizations occur that prevent complex definitions from being transmitted to the Diffusion server many times.

For some types of topic, setting up metadata is part of the task of describing the topic.

Example: Create a JSON topic

The following examples create a JSON topic and receive a stream of values from the topic.

JavaScript

```
diffusion.connect({
    host   : 'diffusion.example.com',
    port   : 443,
    secure : true,
    principal : 'control',
    credentials : 'password'
}).then(function(session) {

    // 1. Data Types are exposed from the top level Diffusion
 namespace. It is often easier
    // to assign these directly to a local variable.
    var jsonDataType = diffusion.datatypes.json();

    // 2. Data Types are currently provided for JSON and Binary topic
 types.
    session.topics.add('topic/json',
 diffusion.topics.TopicType.JSON);

    // 3. Values can be created directly from the data type.
    var jsonValue = jsonDataType.from({
        "foo" : "bar"
```

```
    });

    // Topics are updated using the standard update mechanisms
    session.topics.update('topic/json', jsonValue);

    // Subscriptions are performed normally
    session.select('topic/json');

    // 4. Streams can be specialised to provide values from a
 specific datatype.
    session.addStream('topic/json', jsonDataType).on('value',
 function(topic, specification, newValue, oldValue) {
        // When a JSON or Binary topic is updated, any value handlers
 on a subscription will be called with both the
        // new value, and the old value.

        // The oldValue parameter will be undefined if this is the
 first value received for a topic.

        // For JSON topics, value#get returns a JavaScript object
        // For Binary topics, value#get returns a Buffer instance
        console.log("Update for " + topic, newValue.get());
    });

    // 5. Raw values of an appropriate type can also be used for JSON
 and Binary topics.
    // For example, plain JSON objects can be used to update JSON
 topics.
    session.topics.update('topic/json', {
         "foo" : "baz",
         "numbers" : [1, 2, 3]
    });
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using static System.Console;
```

```
namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Client implementation that adds and updates a JSON topic.
    /// </summary>
    public sealed class JSONTopicsManager {

        public JSONTopicsManager(string serverUrl) {
            var session =
 Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors)
                    => CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            var topicControl = session.TopicControl;
            var topicUpdate = session.TopicUpdate;

            // Create a JSON topic 'random/JSON'
            string topic = "random/JSON";

            try {
                await topicControl.AddTopicAsync(topic,
 TopicType.JSON);
                WriteLine($"Topic '{topic}' added successfully.");
            }
            catch (Exception ex)
            {
                WriteLine( $"Failed to add topic '{topic}' :
 {ex}." );
                session.Close();
                return;
            }

            WriteLine($"Updating topic '{topic}' with a new value:");

            var newValue = Diffusion.DataTypes.JSON.FromJSONString(
                "{\"date\":\"" + DateTime.Today.Date.ToString("D") +
 "\"," +
                "\"time\":\"" + DateTime.Now.TimeOfDay.ToString("g")
 + "\"}");

            try
            {
                await topicUpdate.SetAsync(topic, newValue);

                await Task.Delay(TimeSpan.FromMilliseconds(300));
            }
            catch (Exception ex)
            {
                WriteLine($"Topic {topic} could not be updated :
 {ex}.");
            }

            // Remove the JSON topic 'random/JSON'
            try {
                await topicControl.RemoveTopicsAsync(topic);
            }
            catch (Exception ex)
            {
                WriteLine( $"Failed to remove topic '{topic}' :
 {ex}." );
            }
```

```
            // Close the session
            session.Close();
        }
    }

    /// <summary>
    /// Client implementation that subscribes to a JSON topic and
consumes the data it receives.
    /// </summary>
    public sealed class JSONTopicsConsumer
    {
        public JSONTopicsConsumer(string serverUrl)
        {
            // Connect anonymously
            var session = Diffusion.Sessions.Open(serverUrl);

            // Get the Topics feature to subscribe to topics
            var topics = session.Topics;
            string topic = "random/JSON";

            // Add a topic stream for 'random/JSON'
            var jsonStream = new JSONStream();
            topics.AddStream(topic, jsonStream);

            try
            {
                // Subscribe to 'random/JSON' topic
                await topics.SubscribeAsync(topic);

                await Task.Delay(TimeSpan.FromMilliseconds(300));
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to subscribe to topic '{topic}' :
{ex}.");
            }
            finally
            {
                // Note that closing the session, will automatically
unsubscribe from all topics
                // the client is subscribed to.
                topics.RemoveStream(jsonStream);
                session.Close();
            }
        }

        /// <summary>
        /// Basic implementation of the IValueStream for JSON topics.
        /// </summary>
        private sealed class JSONStream : IValueStream<IJSON>
        {
            /// <summary>
            /// Notification of stream being closed normally.
            /// </summary>
            public void OnClose()
                => WriteLine("The subscrption stream is now
closed.");

            /// <summary>
            /// Notification of a contextual error related to this
callback.
            /// </summary>
```

```
            /// <param name="errorReason">Error reason.</param>
            public void OnError(ErrorReason errorReason)
                => WriteLine($"An error has occured :
 {errorReason}.");

            /// <summary>
            /// Notification of a successful subscription.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            public void OnSubscription(string topicPath,
 ITopicSpecification specification)
                => WriteLine($"Client subscribed to topic
 '{topicPath}'.");

            /// <summary>
            /// Notification of a successful unsubscription.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            /// <param name="reason">Error reason.</param>
            public void OnUnsubscription(string topicPath,
 ITopicSpecification specification,
                                         TopicUnsubscribeReason
 reason)
                => WriteLine($"Client unsubscribed from topic
 '{topicPath}' with reason '{reason}'.");

            /// <summary>
            /// Topic update received.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            /// <param name="oldValue">Value prior to update.</param>
            /// <param name="newValue">Value after update.</param>
            public void OnValue(string topicPath, ITopicSpecification
 specification, IJSON oldValue,
                                IJSON newValue)
                => WriteLine($"New value of topic '{topicPath}' is
 {newValue.ToJSONString()}.");
        }
    }
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import static java.util.Objects.requireNonNull;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Map;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.dataformat.cbor.CBORFactory;
import com.fasterxml.jackson.dataformat.cbor.CBORGenerator;
import com.pushtechnology.diffusion.client.Diffusion;
```

```
import com.pushtechnology.diffusion.client.callbacks.Registration;
import
 com.pushtechnology.diffusion.client.callbacks.TopicTreeHandler;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl.AddCon
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl.Remova
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicUpdateControl;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicUpdateControl.
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicUpdateControl.
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicUpdateControl.
import com.pushtechnology.diffusion.client.session.Session;
import
 com.pushtechnology.diffusion.client.session.SessionClosedException;
import com.pushtechnology.diffusion.client.topics.details.TopicType;
import com.pushtechnology.diffusion.datatype.json.JSON;
import com.pushtechnology.diffusion.datatype.json.JSONDataType;

/**
 * This example shows a control client creating a JSON topic and
 sending updates
 * to it.
 * <P>
 * There will be a topic for each currency for which rates are
 provided. The
 * topic will be created under the FX topic - so, for example FX/GBP
 will
 * contain a map of all rate conversions from the base GBP currency.
 The rates
 * are represented as string decimal values (e.g. "12.457").
 * <P>
 * The {@code addRates} method shows how to create a new rates topic,
 specifying
 * its initial map of values.
 * <P>
 * The {@code changeRates} method which takes a map shows how to
 completely
 * replace the set of rates for a currency with a new map of rates.
 * <P>
 * The {@code changeRates} method which takes a string shows an
 alternative
 * mechanism where the new rates are simply supplied as a JSON
 string.
 * <P>
 * Either of the changeRates methods could be used and after the
 first usage for
 * any topic the values is cached, and so subsequent set calls can
 compare with
 * the last value and send only the differences to the server.
 *
 * @author Push Technology Limited
 * @since 5.7
 * @see ClientConsumingJSONTopics
 */
public final class ControlClientUpdatingJSONTopics {

    private static final String ROOT_TOPIC = "FX";
```

```
   private final Session session;
   private final TopicControl topicControl;
   private volatile TopicUpdateControl.ValueUpdater<JSON>
valueUpdater;
   private volatile Registration updateSourceRegistration;
   private final CBORFactory cborFactory = new CBORFactory();
   private final JSONDataType jsonDataType =
Diffusion.dataTypes().json();

   /**
    * Constructor.
    *
    * @param serverUrl for example "ws://diffusion.example.com:80"
    */
   public ControlClientUpdatingJSONTopics(String serverUrl) {

       cborFactory.setCodec(new ObjectMapper());

       session =

Diffusion.sessions().principal("control").password("password")
               .open(serverUrl);

       topicControl = session.feature(TopicControl.class);

       // Register as an updater for all topics under the root and
request
       // that all topics created are removed when the session
closes

session.feature(TopicUpdateControl.class).registerUpdateSource(
           ROOT_TOPIC,
           new UpdateSource.Default() {
               @Override
               public void onRegistered(
                   String topicPath,
                   Registration registration) {
                   updateSourceRegistration = registration;
               }

               @Override
               public void onActive(String topicPath, Updater
updater) {
                   topicControl.removeTopicsWithSession(
                       ROOT_TOPIC,
                       new TopicTreeHandler.Default());
                   valueUpdater = updater.valueUpdater(JSON.class);
               }

               @Override
               public void onClose(String topicPath) {
                   session.close();
               }
           });

   }

   /**
    * Add a new rates topic.
    *
    * @param currency the base currency
    * @param values the full map of initial rates values
```

```
     * @param callback reports outcome
     * @throws IOException if unable to convert rates map
     */
    public void addRates(
        String currency,
        Map<String, String> values,
        AddContextCallback<String> callback) throws IOException {

        topicControl.addTopic(
            rateTopicName(currency),
            TopicType.JSON,
            mapToJSON(values),
            currency,
            callback);
    }

    /**
     * Update an existing rates topic, replacing the rates mappings
with a new
     * set of mappings.
     *
     * @param currency the base currency
     * @param values the new rates values
     * @param callback reports outcome
     * @throws IOException if unable to convert rates map
     */
    public void changeRates(
        String currency,
        Map<String, String> values,
        UpdateContextCallback<String> callback) throws IOException {

        if (valueUpdater == null) {
            throw new IllegalStateException("Not registered as
updater");
        }

        valueUpdater.update(
            rateTopicName(currency),
            mapToJSON(values),
            currency,
            callback);
    }

    /**
     * Update an existing rates topic, replacing the rates mappings
with a new
     * set of mappings specified as a JSON string, for example
     * {"USD":"123.45","HKD":"456.3"}.
     *
     * @param currency the base currency
     * @param jsonString a JSON string specifying the map of currency
rates
     * @param callback reports the outcome
     * @throws IOException if unable to convert string
     */
    public void changeRates(
        String currency,
        String jsonString,
        UpdateContextCallback<String> callback) throws
SessionClosedException,
        IllegalArgumentException, IOException {

        if (valueUpdater == null) {
```

```
            throw new IllegalStateException("Not registered as
updater");
        }

        valueUpdater.update(
            rateTopicName(currency),
            jsonDataType.fromJsonString(jsonString),
            currency,
            callback);

    }

    /**
     * Convert a given map to a JSON object.
     */
    private JSON mapToJSON(Map<String, String> values) throws
IOException {
        // Use the third-party Jackson library to write out the
values map as a
        // CBOR-format binary.
        final ByteArrayOutputStream baos = new
ByteArrayOutputStream();
        final CBORGenerator generator =
cborFactory.createGenerator(baos);
        generator.writeObject(values);
        return jsonDataType.readValue(baos.toByteArray());
    }

    /**
     * Remove a rates entry (removes its topic) and clear cached
value for the
     * topic.
     *
     * @param currency the currency
     *
     * @param callback reports the outcome
     */
    public void removeRates(
        String currency,
        RemovalContextCallback<String> callback) {

        final String topicName = rateTopicName(currency);

        if (valueUpdater != null) {
            valueUpdater.removeCachedValues(topicName);
        }

        topicControl.remove(topicName, currency, callback);
    }

    /**
     * Close the session.
     */
    public void close() {
        updateSourceRegistration.close();
    }

    /**
     * Generate a hierarchical topic name for a rates topic.
     * <P>
     * e.g. for currency=GBP would return "FX/GBP".
     *
     * @param currency the currency
```

```
     * @return the topic name
     */
    private static String rateTopicName(String currency) {
        return String.format("%s/%s", ROOT_TOPIC,
 requireNonNull(currency));
    }

}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Handling subscriptions to missing topics

A client can handle subscription requests for topics that do not exist and can act on those notifications, for example, by creating a topic on demand that matches the request.

Required permissions: modify_topic, register_handler

The client can register itself as a handler for missing topics for any branch of the topic tree. The client is notified of any subscription using a selector that does not resolve to any existing topics. This enables the client to create the missing topics if your application requires this.

> **Note:** As of Diffusion 6.7, the notification contains the full set of session properties of the requesting session as well as the names of the servers that the request has passed through.

The missing topic handler is removed when the registering session is closed. If the registering session loses connection, it goes into DISCONNECTED state. When in DISCONNECTED state the handler remains active but cannot pass on the notifications to the client. If the client then closes, these notifications are discarded.

To ensure that missing topic notifications are always received by your solution, you can use multiple clients to register missing topic handlers. Ensure that if any of these clients lose connection they go straight to CLOSED state by setting the reconnection timeout to zero so that when the client loses connection it closes straight away, and further missing topic notifications are routed to a handler registered by another client.

Identifying when a particular client connects

You can use missing topic notifications when your application needs to detect when a particular uniquely identifiable client connects.

If a client session subscribes to a topic with a path that does not exist and is unique to that client (for example, the path contains a user ID), this will trigger a missing topic notification, informing any handler that the client has connected.

Registering a missing topic notification handler

Register a handler against a branch of the topic tree:

JavaScript

```
session.topics.addMissingTopicHandler("topic_branch", {
 // Implement handling code
});
```

.NET

```
ITopicControl topicControl  = session.TopicControl;
var missingTopicHandler = new MissingTopicHandler();
topicControl.AddMissingTopicHandler( topic_branch,
 missingTopicHandler );
```

Java and Android

```
TopicControl topicControl = session.feature(TopicControl.class);
topicControl.addMissingTopicHandler("topic_branch", new
 MissingTopicNotificationHandler());
```

C

```
MISSING_TOPIC_PARAMS_T handler = {
        .on_missing_topic = on_missing_topic,
        .topic_path = topic_branch,
        .context = NULL
};

missing_topic_register_handler(session, handler);
```

Apple

```
-(void)registerAsMissingTopicHandlerForSession:(PTDiffusionSession
 *const)session {
    [session.topicControl addMissingTopicHandler:self
                                    forTopicPath:@"topic_branch"

 completionHandler:^(PTDiffusionTopicTreeRegistration *const
 registration, NSError *const error)
    {
        if (registration) {
            NSLog(@"Registered as missing topic handler.");
        } else {
            NSLog(@"Failed to register as missing topic handler.
 Error: %@", error);
        }
    }];
}
```

Related concepts

Using missing topic notifications with fan-out on page 118

Missing topic notifications generated by subscription requests to a secondary server are propagated to missing topic handlers registered against the primary servers.

Example: Receive missing topic notifications

The following examples use the TopicControl feature in the Diffusion API to register a missing topic notification handler.

JavaScript

```
var diffusion = require('diffusion');

// Connect to the server. Change these options to suit your own
 environment.
// Node.js will not accept self-signed certificates by default. If
 you have
// one of these, set the environment variable
 NODE_TLS_REJECT_UNAUTHORIZED=0
// before running this example.
diffusion.connect({
    host   : 'diffusion.example.com',
    port   : 443,
    secure : true
}).then(function(session) {

 // Register a missing topic handler on the "example" root topic
 // Any subscriptions to missing topics along this path will invoke
 this handler
 session.topics.addMissingTopicHandler("example", {
  // Called when a handler is successfully registered
  onRegister : function(path, close) {
   console.log("Registered missing topic handler on path: " + path);
   // Once we've registered the handler, we initiate a subscription
 with the selector "?example/topic/.*"
   // This will invoke the handler.
   session.select("?example/topic/.*");
   session.addStream("?example/topic/.*",
 diffucion.datatypes.string()).on('subscribe', function(path) {
    console.log("Subscribed to topic: " + path);
   });
  },
  // Called when the handler is closed
  onClose : function(path) {
   console.log("Missing topic handler on path '" + path + "' has been
 closed");
  },
  // Called if there is an error on the handler
  onError : function(path, error) {
   console.log("Error on missing topic handler");
  },
  // Called when we've received a missing topic notification on our
 registered handler path
  onMissingTopic : function(notification) {
   console.log("Received missing topic notification with selector: "
 + notification.selector);
   // Once we've received the missing topic notification initiated
 from subscribing to "?example/topic/.*",
   // we add a topic that will match the selector

   var topic = "example/topic/foo";
```

```
   session.topics.add(topic,
 diffusion.topics.TopicType.STRING).then(function(result) {
    console.log("Topic add success: " + topic);
    // If the topic addition is successful, we proceed() with the
 session's subscription.
    // The client will now be subscribed to the topic
    notification.proceed();
   }, function(reason) {
    console.log("Topic add failed: " + reason);
    // If the topic addition fails, we cancel() the session's
 subscription request.
    notification.cancel();
   });
  }
 });
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features.Control.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using static System.Console;

namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Implementation of a missing topic handler.
    /// </summary>
    public sealed class AddMissingTopicHandler
    {
        public AddMissingTopicHandler()
        {
            var controlSession =
 Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
```

```
                .Open(serverUrl);

            var clientSession =
Diffusion.Sessions.Principal("client").Password("password")
                .CertificateValidation((cert, chain, errors) =>
CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            string topicPath = "Example/Some Topic";

            WriteLine($"Adding missing topic handler for topic
'{topicPath}'.");

            var registration =
                await
controlSession.TopicControl.AddMissingTopicHandlerAsync(
                        topicPath,
                        new
MissingTopicNotificationStream(controlSession));

            WriteLine($"Subscribing to topic '{topicPath}'.");

            await clientSession.Topics.SubscribeAsync("?Example/Some
Topic//");

            await Task.Delay(TimeSpan.FromSeconds(1));

            // Clean up
            await session.TopicControl.RemoveTopicsAsync(topicPath);

            WriteLine($"Topic '{topicPath}' removed.");

            await clientSession.Topics.UnsubscribeAsync(selector,
cancellationToken);

            WriteLine($"Unsubscribing to topic '{topicPath}'.");

            await registration.CloseAsync();

            clientSession.Close();
            controlSession.Close();
        }

        /// <summary>
        /// Basic implementation of the stream that will be called
when a session subscribes using
        /// a topic selector that matches no topics.
        /// </summary>
        private sealed class MissingTopicNotificationStream :
IMissingTopicNotificationStream
        {
            private ISession session;

            public MissingTopicNotificationStream(ISession session)
=> this.session = session;

            public void OnClose() => WriteLine("Handler is
removed.");

            public void OnError(ErrorReason errorReason)
                        => WriteLine($"An error has occured :
{errorReason}.");
```

```
            public void OnMissingTopic(IMissingTopicNotification
 notification)
            {
                WriteLine($"Topic '{notification.TopicPath}' does not
 exist.");

                session.TopicControl.AddTopic(
                        notification.TopicPath,

 session.TopicControl.NewSpecification(TopicType.STRING),
                        new TopicControlAddCallback(notification));
            }
        }

        /// <summary>
        /// Implementation of a callback interface for adding topics.
        /// </summary>
        private sealed class TopicControlAddCallback :
 ITopicControlAddCallback
        {
            IMissingTopicNotification notification;

            public TopicControlAddCallback(IMissingTopicNotification
 notification)
                        => this.notification = notification;

            public void OnDiscard() => WriteLine("The stream is now
 closed.");

            public void OnTopicAdded(string topicPath) =>
 WriteLine($"Topic '{topicPath}' added.");

            public void OnTopicAddFailed(string topicPath,
 TopicAddFailReason reason)
                        => WriteLine($"The topic '{topicPath}' could
 not be added - reason: {reason}.");
        }
    }
}
```

Java and Android

```
/
 ************************************************************************
  * Copyright (C) 2014, 2021 Push Technology Ltd.
  *
  * Licensed under the Apache License, Version 2.0 (the "License");
  * you may not use this file except in compliance with the License.
  * You may obtain a copy of the License at
  * http://www.apache.org/licenses/LICENSE-2.0
  *
  * Unless required by applicable law or agreed to in writing,
  software
  * distributed under the License is distributed on an "AS IS" BASIS,
  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  implied.
  * See the License for the specific language governing permissions
  and
  * limitations under the License.
  ************************************************************************
 package com.pushtechnology.diffusion.examples;
```

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.callbacks.ErrorReason;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl.Missin
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl.Missin
import com.pushtechnology.diffusion.client.session.Session;
import com.pushtechnology.diffusion.client.topics.TopicSelector.Type;
import com.pushtechnology.diffusion.client.topics.details.TopicType;

/**
 * An example of registering a missing topic notification handler and
 processing
 * notifications using a control client.
 *
 * @author Push Technology Limited
 */
public final class ControlClientHandlingMissingTopicNotification {

    private static final Logger LOG =

 LoggerFactory.getLogger(ControlClientHandlingMissingTopicNotification.class);

    private final Session session;
    private final TopicControl topicControl;

    /**
     * Constructor.
     */
    public ControlClientHandlingMissingTopicNotification(String
 serverUrl)
        throws InterruptedException, ExecutionException,
 TimeoutException {
        // Create a session
        session =
 Diffusion.sessions().password("password").principal("admin")
            .open(serverUrl);

        topicControl = session.feature(TopicControl.class);

        // Registers a missing topic notification on a topic path
        topicControl.addMissingTopicHandler(
            "Accounts",
            new NotificationStream()).get(5, TimeUnit.SECONDS);

    }

    private final class NotificationStream implements
        MissingTopicNotificationStream {
        @Override
        public void onClose() {
        }

        @Override
```

```
        public void onError(ErrorReason errorReason) {
        }

        @Override
        public void onMissingTopic(MissingTopicNotification
 notification) {
            // This handler will create a missing topic if a path
 selector
            // requesting a topic starting with "Accounts/" is
 selected and
            // the requesting session has the principal 'control'.
            if (notification.getTopicSelector().getType() ==
 Type.PATH) {
                final String path = notification.getTopicPath();
                if (path.startsWith("Accounts/") &&
                    "control".equals(

 notification.getSessionProperties().get(Session.PRINCIPAL))) {

                    topicControl.addTopic(
                        path,
                        TopicType.STRING).whenComplete((result, ex) -
> {
                            if (ex == null) {
                                LOG.info("Missing topic " + path + "
 " + result);
                            }
                            else {
                                LOG.warn("Failed to create missing
 topic " + path, ex);
                            }
                        });
                }
            }
        }
    }

}
```

C

```
/*
 * This example shows how to register a missing topic notification
 * handler and return a missing topic notification response - calling
 * missing_topic_proceed() once we've created the topic.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#include <apr.h>
#include <apr_thread_mutex.h>
#include <apr_thread_cond.h>

#include "diffusion.h"
#include "args.h"

ARG_OPTS_T arg_opts[] = {
        ARG_OPTS_HELP,
```

```
        {'u', "url", "Diffusion server URL", ARG_OPTIONAL,
 ARG_HAS_VALUE, "ws://localhost:8080"},
        {'p', "principal", "Principal (username) for the connection",
 ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'c', "credentials", "Credentials (password) for the
 connection", ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'r', "topic_root", "Topic root to process missing topic
 notifications on", ARG_OPTIONAL, ARG_HAS_VALUE, "foo"},
        END_OF_ARG_OPTS
};

static int
on_topic_added(SESSION_T *session, const SVC_ADD_TOPIC_RESPONSE_T
 *response, void *context)
{
        puts("Topic added");
        return HANDLER_SUCCESS;
}

static int
on_topic_add_failed(SESSION_T *session, const
 SVC_ADD_TOPIC_RESPONSE_T *response, void *context)
{
        puts("Topic add failed");
        printf("Reason code: %d\n", response->reason);
        return HANDLER_SUCCESS;
}

static int
on_topic_add_discard(SESSION_T *session, void *context)
{
        puts("Topic add discarded");
        return HANDLER_SUCCESS;
}

/*
 * A request has been made for a topic that doesn't exist; create it
 * and inform Diffusion that the client's subcription request can
 * proceed.
 */
static int
on_missing_topic(SESSION_T *session, const
 SVC_MISSING_TOPIC_REQUEST_T *request, void *context)
{
        printf("Missing topic: %s\n", request->topic_selector);

        BUF_T *sample_data_buf = buf_create();
        buf_write_string(sample_data_buf, "Hello, world");

        // Add the missing topic.
        ADD_TOPIC_PARAMS_T topic_params = {
                .on_topic_added = on_topic_added,
                .on_topic_add_failed = on_topic_add_failed,
                .on_discard = on_topic_add_discard,
                .topic_path = strdup(request->topic_selector+1),
                .details =
 create_topic_details_single_value(M_DATA_TYPE_STRING),
                .content = content_create(CONTENT_ENCODING_NONE,
 sample_data_buf)
        };

        add_topic(session, topic_params);
```

```
        // Proceed with the client's subscription to the topic
        missing_topic_proceed(session, (SVC_MISSING_TOPIC_REQUEST_T
 *) request);

        return HANDLER_SUCCESS;
}

/*
 * Entry point for the example.
 */
int
main(int argc, char **argv)
{
        /*
         * Standard command-line parsing.
         */
        HASH_T *options = parse_cmdline(argc, argv, arg_opts);
        if(options == NULL || hash_get(options, "help") != NULL) {
                show_usage(argc, argv, arg_opts);
                return EXIT_FAILURE;
        }

        const char *url = hash_get(options, "url");
        const char *principal = hash_get(options, "principal");
        const char *topic_root = hash_get(options, "topic_root");

        CREDENTIALS_T *credentials = NULL;
        const char *password = hash_get(options, "credentials");
        if(password != NULL) {
                credentials = credentials_create_password(password);
        }

        SESSION_T *session;
        DIFFUSION_ERROR_T error = { 0 };

        session = session_create(url, principal, credentials, NULL,
 NULL, &error);
        if(session != NULL) {
                printf("Session created (state=%d, id=%s)\n",
                       session_state_get(session),
                       session_id_to_string(session->id));
        }
        else {
                printf("Failed to create session: %s\n",
 error.message);
                free(error.message);
                return EXIT_FAILURE;
        }

        /*
         * Register the missing topic handler
         */
        MISSING_TOPIC_PARAMS_T handler = {
                .on_missing_topic = on_missing_topic,
                .topic_path = topic_root,
                .context = NULL
        };

        missing_topic_register_handler(session, handler);

        /*
         * Run for 5 minutes.
         */
```

```
        sleep(5 * 60);

        /*
         * Close session and clean up.
         */
        session_close(session, NULL);
        session_free(session);

        hash_free(options, NULL, free);

        return EXIT_SUCCESS;
}
```

Apple

```
//  Diffusion Client Library for iOS and OS X - User Manual Code
 Snippets
//
//  Copyright (C) 2021 Push Technology Ltd.
//
//  Licensed under the Apache License, Version 2.0 (the "License");
//  you may not use this file except in compliance with the License.
//  You may obtain a copy of the License at
//  http://www.apache.org/licenses/LICENSE-2.0
//
//  Unless required by applicable law or agreed to in writing,
 software
//  distributed under the License is distributed on an "AS IS" BASIS,
//  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
//  See the License for the specific language governing permissions
 and
//  limitations under the License.

import Foundation
import Diffusion

class MissingTopicHandlerExample: PTDiffusionMissingTopicHandler  {
    var session: PTDiffusionSession?

    func startWithURL(url: URL) {
        let credentials = PTDiffusionCredentials(password:
 "password")

        let sessionConfiguration =
            PTDiffusionSessionConfiguration(principal: "control",
                                            credentials:credentials)

        print("Connecting...")

        PTDiffusionSession.open(with: url,
                                configuration: sessionConfiguration)
 { (session, error) in

            if (session == nil) {
                print("Failed to open session: %@",
 error!.localizedDescription)
                return
            }

            // At this point we now have a connected session.
```

```
            print("Connected.")

            // To maintain a strong reference to the session.
            self.session = session!

            self.registerAsMissingTopicHandler(session: session!)
        }
    }

    func registerAsMissingTopicHandler(session: PTDiffusionSession) {
        session.topicControl.add(self, forTopicPath: "Example/Control
Client Handler") { (registration, error) in

            if (registration != nil) {
                print("Registered as missing topic handler.")
            }
            else {
                print("Failed to register as missing topic handler:
%@", error!.localizedDescription)
            }
        }
    }

    func diffusionTopicTreeRegistration(_ registration:
PTDiffusionTopicTreeRegistration,
                                        hadMissingTopicNotification
notification: PTDiffusionMissingTopicNotification) {
        print("Received Missing Topic Notification: %@",
notification);

        let expression: String = notification.topicSelectorExpression

        // Expect a path pattern expression.
        if (!expression.hasPrefix(">")) {
            print("Topic selector expression is not a path pattern.")
            return
        }

        // extract topic path from path pattern expression
        let index = expression.index(expression.startIndex, offsetBy:
1)
        let topicPath = String(expression[index...])

        // Add a stateless topic at this topic path.
        self.session?.topicControl.addTopic(withPath: topicPath,

type:PTDiffusionTopicType.string,
                                            completionHandler:
{ (result, error) in
            if (result == nil) {
                print("Error occurred while creating topic: %@",
error!.localizedDescription)
            }
            else {
                print("Topic created.")
            }
        })
    }

    func diffusionTopicTreeRegistrationDidClose(_ registration:
PTDiffusionTopicTreeRegistration) {
        print("Registration %@ closed.")
    }
```

```
    func diffusionTopicTreeRegistration(_ registration:
 PTDiffusionTopicTreeRegistration, didFailWithError error: Error) {
        print("Registration %@ failed: %@", registration,
 error.localizedDescription)
    }

}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Defining a recordV2 schema

You can use the API to specify a schema that defines the content of a recordV2 topic.

About this task

Publishing clients can define an optional schema for a recordV2 topic. The topic value must conform to the schema.

No client session is required to create a schema.

The Diffusion API for the following platforms provides builder methods that enable you to define a schema:

- Java
- Android

The following example demonstrates how to create a schema using the Java API.

Procedure

1. Import the required classes.

```
import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.datatype.recordv2.RecordV2DataType;
import
 com.pushtechnology.diffusion.datatype.recordv2.schema.Schema;
import
 com.pushtechnology.diffusion.datatype.recordv2.schema.SchemaBuilder;
```

2. Create an example class with a recordV2 datatype and a constructor.

```
public final class ClientCreatingRecordV2Schema {

    private final RecordV2DataType dataType =
 Diffusion.dataTypes().recordV2();

    /**
    * Constructor.
    */
    public ClientCreatingRecordV2Schema() {
    }
}
```

3. Create a schema.

This schema specifies that the topic will contain a record containing a field that can occur from two to five times, and a different record with a field that can occur unlimited times.

```
        public Schema createVariableRepeatingFieldsSchema() {
        final SchemaBuilder builder = dataType.schemaBuilder();
        return builder
            .record("A").string("repeatingField", 2, 5)
            .record("B").string("repeatingFieldUnlimited", 1, -1)
            .build();
    }
```

See the full example code below for how to construct a variety of different schemas.

For more information, see Java API documentation.

Example: A class that shows how to create different schemas.

```
/
*******************************************************************************
 * Copyright (C) 2017 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the
 License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS"
 BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 *******************************************************************************
package com.pushtechnology.diffusion.examples;

import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.datatype.recordv2.RecordV2DataType;
import
 com.pushtechnology.diffusion.datatype.recordv2.schema.Schema;
import
 com.pushtechnology.diffusion.datatype.recordv2.schema.SchemaBuilder;

/**
 * This example class has a number of methods that demonstrate the
 creation of
 * schemas for RECORD_V2 topics, using the Diffusion Client API.
 * <P>
 * Note that no client session is required in order to create a
 schema.
 *
 * @author Push Technology Limited
 * @since 6.0
 */
public final class ClientCreatingRecordV2Schema {
```

```
   private final RecordV2DataType dataType =
Diffusion.dataTypes().recordV2();

   /**
    * Constructor.
    */
   public ClientCreatingRecordV2Schema() {
   }

   /**
    * Example of a schema consisting of a single record with
three fields each
    * of s different data type.
    *
    * @return a schema
    */
   public Schema createSimpleSchema() {
       final SchemaBuilder builder = dataType.schemaBuilder();
       return builder
           .record("Record")

.string("string").integer("integer").decimal("decimal", 3)
           .build();
   }

   /**
    * Example of a schema consisting of multiple records, each
record with a
    * single field of a specific type.
    *
    * @return a schema
    */
   public Schema createMultipleRecordsSchema() {
       final SchemaBuilder builder = dataType.schemaBuilder();
       return builder
           .record("StringRecord").string("string")
           .record("IntegerRecord").integer("integer")
           .record("DecimalRecord").decimal("decimal", 3)
           .build();
   }

   /**
    * Example of a schema consisting of a record (with a single
string field)
    * repeating exactly 10 times.
    *
    * @return a schema
    */
   public Schema createFixedRepeatingRecordsSchema() {
       final SchemaBuilder builder = dataType.schemaBuilder();
       return builder
           .record("RepeatingRecord", 10).string("string")
           .build();
   }

   /**
    * Example of a schema consisting of 2 record types.
"FixedRecord" is a
    * record that occurs 5 times. "RepeatingRecord" is an
optional record that
    * can be repeated as many times as required (unlimited).
    *
    * @return a schema
```

```
     */
    public Schema createVariableRepeatingRecordsSchema() {
        final SchemaBuilder builder = dataType.schemaBuilder();
        return builder
            .record("FixedRecord", 5).string("a")
            .record("RepeatingRecord", 0, -1).string("b")
            .build();
    }

    /**
     * Example of a schema consisting of a single record with a
string field
     * that occurs exactly 10 times.
     *
     * @return a schema
     */
    public Schema createFixedRepeatingFieldsSchema() {
        final SchemaBuilder builder = dataType.schemaBuilder();
        return builder
            .record("Record").string("repeatingString", 10)
            .build();
    }

    /**
     * Example of a schema consisting of two records. The first
record (A) has a
     * field, "repeatingField", which can occur between 2 and 5
times. The
     * second record (B) has a field, "repeatingFieldUnlimited",
which can occur
     * as many times as required but at least once.
     *
     * @return a schema
     */
    public Schema createVariableRepeatingFieldsSchema() {
        final SchemaBuilder builder = dataType.schemaBuilder();
        return builder
            .record("A").string("repeatingField", 2, 5)
            .record("B").string("repeatingFieldUnlimited", 1, -1)
            .build();
    }

    /**
     * Example of a schema consisting of a single record and
multiple fields
     * encapsulating a person's name and address.
     *
     * @return a schema
     */
    public Schema createNameAndAddressSchema() {
        final SchemaBuilder builder = dataType.schemaBuilder();
        return builder
            .record("nameAndAddress")
            .string("firstName")
            .string("surname")
            .integer("houseNumber")
            .string("street")
            .string("town")
            .string("state")
            .string("postCode")
            .build();
    }
```

```
}
```

Related concepts

RecordV2 topics on page 89
A topic that streams data in recordV2 format, where the data is divided into multiple records, each of which can contain multiple fields. RecordV2 topics are stateful: each topic stores a value consisting of one or more records on the Diffusion server.

RecordV2 schema on page 91
A schema is an optional way to define how data is formatted when it is published on a recordV2 topic. A schema defines and names the permitted records and fields within the topic, and enables direct access to the fields.

Update recordV2 topics on page 270
The following example demonstrates how to create and update recordV2 topics, including the use of a schema.

Subscribe to recordV2 topics on page 275
The following example demonstrates how to process information from subscribed recordV2 topics, including the use of a schema.

Update recordV2 topics

The following example demonstrates how to create and update recordV2 topics, including the use of a schema.

This example demonstrates a Java control client updating recordV2 topics containing currency exchange rate information.

Each topic contains a record with two decimal fields, representing the buy and sell rates between a pair of currencies.

The example can be run either with or without a schema.

Java and Android

```
/
*******************************************************************************
 * Copyright (C) 2017, 2019 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 *******************************************************************************
package com.pushtechnology.diffusion.examples;

import static
 com.pushtechnology.diffusion.client.Diffusion.newTopicSpecification;
```

```
import static
 com.pushtechnology.diffusion.client.topics.details.TopicSpecification.REMOVAL;
import static
 com.pushtechnology.diffusion.client.topics.details.TopicSpecification.SCHEMA;
import static
 com.pushtechnology.diffusion.client.topics.details.TopicType.RECORD_V2;
import static
 com.pushtechnology.diffusion.client.topics.details.TopicType.STRING;
import static java.util.concurrent.TimeUnit.SECONDS;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeoutException;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.features.TopicUpdate;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl;
import com.pushtechnology.diffusion.client.session.Session;
import
 com.pushtechnology.diffusion.client.topics.details.TopicSpecification;
import com.pushtechnology.diffusion.datatype.recordv2.RecordV2;
import
 com.pushtechnology.diffusion.datatype.recordv2.RecordV2DataType;
import
 com.pushtechnology.diffusion.datatype.recordv2.model.MutableRecordModel;
import com.pushtechnology.diffusion.datatype.recordv2.schema.Schema;

/**
 * An example of using a control client to create and update a
 RecordV2 topic in
 * exclusive mode.
 * <P>
 * This uses the 'TopicControl' feature to create a topic and the
 * 'TopicUpdate' feature to send updates to it.
 * <P>
 * To send updates to a topic, the client session requires the
 'update_topic'
 * permission for that branch of the topic tree.
 * <P>
 * The example can be used with or without the use of a schema. This
 is simply
 * to demonstrate the different mechanisms and is not necessarily
 demonstrating
 * the most efficient way to update such a topic.
 *
 * @author Push Technology Limited
 * @since 6.0
 * @see ClientConsumingRecordV2Topics
 */
public final class ControlClientUpdatingRecordV2Topics {

    private static final String ROOT_TOPIC = "FX";

    private final Session session;
    private final TopicControl topicControl;
    private final TopicSpecification topicSpecification;
    private final Schema schema;
    private final RecordV2DataType dataType;

    /**
     * Constructor.
     *
```

```
     * @param serverUrl for example "ws://diffusion.example.com:80"
     */
    public ControlClientUpdatingRecordV2Topics(
        String serverUrl,
        boolean withSchema)
            throws InterruptedException, ExecutionException,
TimeoutException {

        session =

Diffusion.sessions().principal("control").password("password")
                .open(serverUrl);

        topicControl = session.feature(TopicControl.class);

        // Create the root topic that will remove itself when the
session closes
        final TopicSpecification specification =
            newTopicSpecification(STRING)
                .withProperty(
                    REMOVAL,
                    "when this session closes remove '?" + ROOT_TOPIC
+ "//'");

        topicControl.addTopic(ROOT_TOPIC, specification).get(5,
SECONDS);

        dataType = Diffusion.dataTypes().recordV2();

        if (withSchema) {
            schema = dataType.schemaBuilder()
                .record("Rates").decimal("Bid", 5).decimal("Ask",
5).build();
            // Create the topic specification to be used for all
rates topics
            topicSpecification =
                newTopicSpecification(RECORD_V2)
                    .withProperty(
                        SCHEMA,
                        schema.asJSONString());
        }
        else {
            schema = null;
            // Create the topic specification to be used for all
rates topics
            topicSpecification =
                newTopicSpecification(RECORD_V2);
        }
    }

    /**
     * Adds a new conversion rate in terms of base currency and
target currency.
     *
     * The bid and ask rates are entered as strings which may be a
decimal value
     * which will be parsed and validated, rounding to 5 decimal
places.
     *
     * @param currency the base currency (e.g. GBP)
     *
     * @param targetCurrency the target currency (e.g. USD)
     */
```

```
    public void addRateTopic(
        String currency,
        String targetCurrency)
        throws InterruptedException, ExecutionException,
TimeoutException {

        topicControl.addTopic(
            rateTopicName(currency, targetCurrency),
            topicSpecification).get(5, SECONDS);
    }

    /**
     * Set a rate.
     * <P>
     * The rate topic in question must have been added first using
     * {@link #addRateTopic} otherwise this will fail.
     *
     * @param currency the base currency
     *
     * @param targetCurrency the target currency
     *
     * @param bid the new bid rate
     *
     * @param ask the new ask rate
     * @return a CompletableFuture that completes when a response is
received
     *         from the server
     */
    public CompletableFuture<?> setRate(
        String currency,
        String targetCurrency,
        String bid,
        String ask) {

        final RecordV2 value;
        if (schema == null) {
            value = dataType.valueBuilder().addFields(bid,
ask).build();
        }
        else {
            // Mutable models could be kept and reused but for this
simple
            // example one is created every time
            final MutableRecordModel model =
                schema.createMutableModel();
            model.set("Bid", bid);
            model.set("Ask", ask);
            value = model.asValue();
        }

        return session.feature(TopicUpdate.class).set(
            rateTopicName(currency, targetCurrency),
            RecordV2.class,
            value);
    }

    /**
     * Remove a rate (removes its topic).
     *
     * @param currency the base currency
     *
     * @param targetCurrency the target currency
     */
```

```
    public void removeRate(
        String currency,
        String targetCurrency)
        throws InterruptedException, ExecutionException,
 TimeoutException {

        topicControl.removeTopics(
            rateTopicName(currency, targetCurrency))
            .get(5, SECONDS);
    }

    /**
     * Removes a currency (removes its topic and all subordinate rate
 topics).
     *
     * @param currency the base currency
     */
    public void removeCurrency(String currency)
        throws InterruptedException, ExecutionException,
 TimeoutException {
        topicControl
            .removeTopics(String.format("?%s/%s//", ROOT_TOPIC,
 currency))
            .get(5, SECONDS);
    }

    /**
     * Close the session.
     */
    public void close() throws InterruptedException {
        session.close();
    }

    /**
     * Generates a hierarchical topic name for a rate topic.
     * <P>
     * e.g. for currency=GBP and targetCurrency=USD would return "FX/
GBP/USD".
     *
     * @param currency the base currency
     * @param targetCurrency the target currency
     * @return the topic name
     */
    private static String rateTopicName(String currency,
        String targetCurrency) {
        return String.format("%s/%s/%s", ROOT_TOPIC, currency,
 targetCurrency);
    }

}
```

Related concepts

RecordV2 topics on page 89
A topic that streams data in recordV2 format, where the data is divided into multiple records, each of which can contain multiple fields. RecordV2 topics are stateful: each topic stores a value consisting of one or more records on the Diffusion server.

RecordV2 schema on page 91

A schema is an optional way to define how data is formatted when it is published on a recordV2 topic. A schema defines and names the permitted records and fields within the topic, and enables direct access to the fields.

Subscribe to recordV2 topics on page 275
The following example demonstrates how to process information from subscribed recordV2 topics, including the use of a schema.

Related tasks

Defining a recordV2 schema on page 266
You can use the API to specify a schema that defines the content of a recordV2 topic.

Subscribe to recordV2 topics

The following example demonstrates how to process information from subscribed recordV2 topics, including the use of a schema.

This example demonstrates a Java client consuming recordV2 topics which contain currency conversion rates.

Each topic contains a record with two decimal fields, representing the buy and sell rates between a pair of currencies.

The example can be run either with or without a schema.

Java and Android

```
/
*******************************************************************************
 * Copyright (C) 2017 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 *******************************************************************************
package com.pushtechnology.diffusion.examples;

import static java.util.Objects.requireNonNull;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.features.Topics;
import
 com.pushtechnology.diffusion.client.features.Topics.UnsubscribeReason;
```

```
import
 com.pushtechnology.diffusion.client.features.Topics.ValueStream;
import com.pushtechnology.diffusion.client.session.Session;
import
 com.pushtechnology.diffusion.client.topics.details.TopicSpecification;
import com.pushtechnology.diffusion.datatype.recordv2.RecordV2;
import com.pushtechnology.diffusion.datatype.recordv2.RecordV2Delta;
import
 com.pushtechnology.diffusion.datatype.recordv2.RecordV2Delta.Change;
import
 com.pushtechnology.diffusion.datatype.recordv2.model.RecordModel;
import com.pushtechnology.diffusion.datatype.recordv2.schema.Schema;
import
 com.pushtechnology.diffusion.datatype.recordv2.schema.SchemaParseException;

/**
 * This demonstrates a client consuming RecordV2 topics.
 * <P>
 * It has been contrived to demonstrate the various techniques for
 Diffusion
 * record topics and is not necessarily realistic or efficient in its
 * processing.
 * <P>
 * It can be run using a schema or not using a schema and
 demonstrates how the
 * processing could be done in both cases.
 * <P>
 * This makes use of the 'Topics' feature only.
 * <P>
 * To subscribe to a topic, the client session must have the
 'select_topic' and
 * 'read_topic' permissions for that branch of the topic tree.
 * <P>
 * This example receives updates to currency conversion rates via a
 branch of
 * the topic tree where the root topic is called "FX" which under it
 has a topic
 * for each base currency and under each of those is a topic for each
 target
 * currency which contains the bid and ask rates. So a topic FX/GBP/
USD would
 * contain the rates for GBP to USD.
 * <P>
 * This example maintains a local map of the rates and also notifies
 a listener
 * of any rates changes.
 *
 * @author Push Technology Limited
 * @since 6.0
 * @see ControlClientUpdatingRecordV2Topics
 */
public final class ClientConsumingRecordV2Topics {

    private static final Logger LOG =
        LoggerFactory.getLogger(ClientConsumingRecordV2Topics.class);

    private static final String ROOT_TOPIC = "FX";

    /**
     * The map of currency codes to currency objects which each
 maintain rates
     * for each target currency.
     */
```

```
   private final Map<String, Currency> currencies = new
ConcurrentHashMap<>();

   private Schema schema;

   private final RatesListener listener;

   private final Session session;

   /**
    * Constructor.
    *
    * @param serverUrl for example "ws://diffusion.example.com:80"
    * @param listener a listener that will be notified of all rates
and rate
    *        changes
    */
   public ClientConsumingRecordV2Topics(String serverUrl,
       RatesListener listener) {

       this.listener = requireNonNull(listener);

       session =

Diffusion.sessions().principal("client").password("password")
               .open(serverUrl);

       // Use the Topics feature to add a record value stream and
subscribe to
       // all topics under the root.
       final Topics topics = session.feature(Topics.class);
       final String topicSelector = String.format("?%s//",
ROOT_TOPIC);

       topics.addStream(
           topicSelector,
           RecordV2.class,
           new RatesValueStream());

       topics.subscribe(topicSelector)
           .whenComplete((voidResult, exception) -> {
               if (exception != null) {
                   LOG.info("subscription failed", exception);
               }
           });
   }

   /**
    * Returns the rates for a given base and target currency.
    *
    * @param currency the base currency
    * @param targetCurrency the target currency
    * @return the rates or null if there is no such base or target
currency
    */
   public Rates getRates(String currency, String targetCurrency) {
       final Currency currencyObject = currencies.get(currency);
       if (currencyObject != null) {
           return currencyObject.getRates(targetCurrency);
       }
       return null;
   }
```

```
    /**
     * This is used to apply topic stream updates to the local map
and notify
     * listener of changes.
     */
    private void applyUpdate(
        String currency,
        String targetCurrency,
        RecordV2 oldValue,
        RecordV2 newValue) {

        Currency currencyObject = currencies.get(currency);
        if (currencyObject == null) {
            currencyObject = new Currency();
            currencies.put(currency, currencyObject);
        }

        if (schema == null) {
            updateWithoutSchema(
                currency,
                targetCurrency,
                oldValue,
                newValue,
                currencyObject);
        }
        else {
            updateWithSchema(
                currency,
                targetCurrency,
                oldValue,
                newValue,
                currencyObject);
        }
    }

    private void updateWithSchema(
        String currency,
        String targetCurrency,
        RecordV2 oldValue,
        RecordV2 newValue,
        Currency currencyObject) {

        // A data model is generated using the schema allowing direct
access to
        // the fields within it
        final RecordModel model = newValue.asModel(schema);
        final String bid = model.get("Bid");
        final String ask = model.get("Ask");

        currencyObject.setRate(targetCurrency, bid, ask);

        if (oldValue == null) {
            listener.onNewRate(currency, targetCurrency, bid, ask);
        }
        else {
            // Generate a structural delta to determine what has
changed
            final RecordV2Delta delta = newValue.diff(oldValue);
            for (Change change : delta.changes(schema)) {
                final String fieldName = change.fieldName();
                listener.onRateChange(
                    currency,
                    targetCurrency,
```

```
                    fieldName,
                    model.get(fieldName));
            }
        }
    }

    private void updateWithoutSchema(
        String currency,
        String targetCurrency,
        RecordV2 oldValue,
        RecordV2 newValue,
        Currency currencyObject) {

        // All of the fields in the value are obtained.
        final List<String> fields = newValue.asFields();
        final String bid = fields.get(0);
        final String ask = fields.get(1);

        currencyObject.setRate(targetCurrency, bid, ask);

        if (oldValue == null) {
            listener.onNewRate(currency, targetCurrency, bid, ask);
        }
        else {
            // Fields in the old value are obtained to determine what
has
            // changed
            final List<String> oldfields = oldValue.asFields();
            final String oldBid = oldfields.get(0);
            final String oldAsk = oldfields.get(1);
            if (!bid.equals(oldBid)) {
                listener.onRateChange(currency, targetCurrency,
"Bid", bid);
            }
            if (!ask.equals(oldAsk)) {
                listener.onRateChange(currency, targetCurrency,
"Ask", ask);
            }
        }
    }

    private void removeCurrency(String currency) {
        final Currency oldCurrency = currencies.remove(currency);
        for (String targetCurrency : oldCurrency.rates.keySet()) {
            listener.onRateRemoved(currency, targetCurrency);
        }
    }

    private void removeRate(
        String currency,
        String targetCurrency) {

        final Currency currencyObject = currencies.get(currency);
        if (currencyObject != null) {
            if (currencyObject.rates.remove(targetCurrency) != null)
{
                listener.onRateRemoved(currency, targetCurrency);
            }
        }
    }

    /**
     * Close session.
```

```
     */
    public void close() {
        currencies.clear();
        session.close();
    }

    /**
     * Encapsulates a base currency and all of its known rates.
     */
    private static class Currency {

        private final Map<String, Rates> rates = new HashMap<>();

        private Rates getRates(String currency) {
            return rates.get(currency);
        }

        private void setRate(String currency, String bid, String ask)
{
            rates.put(currency, new Rates(bid, ask));
        }

    }

    /**
     * Encapsulates the rates for a particular base/target currency
pair.
     */
    public static final class Rates {

        private final String bidRate;
        private final String askRate;

        /**
         * Constructor.
         *
         * @param bid the bid rate or ""
         * @param ask the ask rate or ""
         */
        private Rates(String bid, String ask) {
            bidRate = bid;
            askRate = ask;
        }

        /**
         * Returns the bid rate.
         *
         * @return bid rate or "" if not available
         */
        public String getBidRate() {
            return bidRate;
        }

        /**
         * Returns the ask rate.
         *
         * @return ask rate or "" if not available
         */
        public String getAskRate() {
            return askRate;
        }

    }
```

```
   /**
    * A listener for Rates updates.
    */
   public interface RatesListener {

       /**
        * Notification of a new rate or rate update.
        *
        * @param currency the base currency
        * @param targetCurrency the target currency
        * @param bid rate
        * @param ask rate
        */
       void onNewRate(String currency, String targetCurrency, String
bid,
           String ask);

       /**
        * Notification of a change to the bid or ask value for a
rate.
        *
        * @param currency the base currency
        * @param targetCurrency the target currency
        * @param bidOrAsk "Bid" or "Ask"
        * @param rate the new rate
        */
       void onRateChange(String currency, String targetCurrency,
           String bidOrAsk, String rate);

       /**
        * Notification of a rate being removed.
        *
        * @param currency the base currency
        * @param targetCurrency the target currency
        */
       void onRateRemoved(String currency, String targetCurrency);
   }

   private final class RatesValueStream
       extends ValueStream.Default<RecordV2> {

       @Override
       public void onSubscription(String topicPath,
TopicSpecification specification) {
           final boolean isRatesTopic =
Diffusion.topicSelectors().parse("?FX/.*/.*").selects(topicPath);
           // Only retrieve a schema when subscribing to a rates
topic
           if (isRatesTopic) {
               final String schemaString =
specification.getProperties().get(TopicSpecification.SCHEMA);
               // If a schema is provided on subscription, retrieve
it and set it once
               // All schemas are identical for rates topics.
               if (schemaString != null && schema == null) {
                   try {
                       schema =
Diffusion.dataTypes().recordV2().parseSchema(schemaString);
                   }
                   catch (SchemaParseException e) {
                       LOG.error("Unable to parse recordV2 schema",
e);
```

```
                    }
                }
            }
        }

        @Override
        public void onValue(String topicPath, TopicSpecification
specification,
            RecordV2 oldValue, RecordV2 newValue) {
            final String[] topicElements = elements(topicPath);
            // It is only a rate update if topic has 2 elements below
root path
            if (topicElements.length == 2) {
                applyUpdate(
                    topicElements[0], // The base currency
                    topicElements[1], // The target currency
                    oldValue,
                    newValue);
            }
        }

        @Override
        public void onUnsubscription(String topicPath,
            TopicSpecification specification, UnsubscribeReason
reason) {
            final String[] topicElements = elements(topicPath);
            if (topicElements.length == 2) {
                removeRate(topicElements[0], topicElements[1]);
            }
            else if (topicElements.length == 1) {
                removeCurrency(topicElements[0]);
            }
        }

        private String[] elements(String topicPath) {
            final String subPath =
                topicPath.replaceFirst("^" + ROOT_TOPIC + "/", "");
            return subPath.split("/");
        }
    }

}
```

Related concepts

RecordV2 topics on page 89
A topic that streams data in recordV2 format, where the data is divided into multiple records, each of which can contain multiple fields. RecordV2 topics are stateful: each topic stores a value consisting of one or more records on the Diffusion server.

RecordV2 schema on page 91
A schema is an optional way to define how data is formatted when it is published on a recordV2 topic. A schema defines and names the permitted records and fields within the topic, and enables direct access to the fields.

Update recordV2 topics on page 270
The following example demonstrates how to create and update recordV2 topics, including the use of a schema.

Related tasks

Defining a recordV2 schema on page 266

You can use the API to specify a schema that defines the content of a recordV2 topic.

Removing topics

A client can use the TopicControl feature of the Diffusion API to add and remove topics at the server.

Required permissions: modify_topic

Currently all topics created using a client have a lifespan the same as the Diffusion server (unless persistence is enabled). The topics remain at the Diffusion server even after the client session that created them has closed unless you explicitly specify that the topic is removed with the session.

A client can remove topics anywhere in the topic tree. The remove operation takes a topic selector, which enables the client to remove many topics at once.

You can also remove all topics beneath a selected topic path by appending the descendant pattern qualifiers, / and //.

Only topics for which the client has modify_topic permission are removed.

If there are topics for which the client does not have modify_topic permission, they are unaffected and the operation completes without throwing an exception.

A client cannot remove topics created by a publisher.

A publisher cannot remove topics created by a client.

For more information, see Topic selectors on page 44.

JavaScript

```
session.topics.removeSelector('topic_selector'))
   .then(function() {
       console.log('Removed all topics that match the selector.');
   });
```

.NET

```
ITopicControl topicControl = session.TopicControl;
topicControl.RemoveTopics( topic_selector, callback );
```

Java and Android

```
TopicControl topicControl = session.feature(TopicControl.class);
topicControl.remove(topic_selector, callback);
```

C

```
// Define callbacks elsewhere
REMOVE_TOPICS_PARAMS_T remove_params = {
        .on_removed = on_topic_removed,
        .on_discard = on_topic_remove_discard,
        .topic_selector = "topic_selector"
};
```

```
remove_topics(session, remove_params);
```

Apple

```
// Remove topic.
[session.topicControl removeDiscreteWithTopicSelectorExpression:
 topic_selector

 completionHandler:^(NSError *const error)
{
    if (error) {
        NSLog(@"Failed to remove topic. Error: %@", error);
    } else {
        NSLog(@"Topic removal request succeeded.");
    }
}];
```

Removing topics automatically

You can specify an automatic removal policy that specifies under what conditions that topic and/or other topics will be removed automatically.

Specifying automatic removal policies

The automatic removal policy for a topic is specified using the REMOVAL topic property. The property is specified as an expression which defines one or more conditions that are to be satisfied before automatic removal occurs, and an optional clause that specifies which topics to remove.

The format of the expression is:

`when` **`conditions`** `[remove "`**`selector`**`"]`

where:

- **conditions** is one or more of the condition types in the table below, separated by `and` or `or` logical operators.
- the `remove` clause is optional. If not added, only the topic with the removal policy will be removed.
- **selector** is a `TopicSelector` expression representing a set of topics to be removed. If a remove clause is specified, the topic with the removal policy will only be removed if its path matches the selector expression.

Table 32: Removal condition types

Condition type	Format	Usage
time after	`time after` *`absoluteTime`*	Removal should occur after a specified absolute time. Absolute time may be specified as a number of milliseconds since 00:00:00 on 1 January 1970 UTC, or as a quoted date and time formatted in RFC_1123 (`"Sun, 3 Jun 2018 11:05:30 GMT"`) date

Condition type	Format	Usage
		time format. Either single or double quotes may be used.
subscriptions less than	`subscriptions < n for forPeriod [after afterPeriod]`	Removal should occur when the topic has had less than the specified number (n) of subscriptions for a given period (forPeriod) of time. Optionally, an initial period (afterPeriod) may be specified by which to delay the initial checking of this condition. See below for period formats.
local subscriptions less than	`local subscriptions < n for forPeriod [after afterPeriod]`	Behaves the same as `subscriptions less than`, but excludes subscriptions via fan-out connections.
no updates for	`no updates for forPeriod [after afterPeriod]`	Removal should occur when the topic has had no updates for a given period (forPeriod) of time. Optionally, an initial period (afterPeriod) may be specified by which to delay the initial checking of this condition. See below for period formats.
no session has	`no session has "criteria" [for forPeriod] [after afterPeriod]`	Removal should occur when there are no sessions satisfying the specified criteria. Optionally, the criteria can be required to be satisfied for a period of time (forPeriod). Optionally, an initial period (afterPeriod) can be specified to delay the initial check of the criteria. Session selection criteria are specified as defined in Session filtering on page 222 and must be surrounded by single or double quotes. See below for period formats. Can have multiple `no session has` clauses in a policy.
no local session has	`no local session has "criteria" [for forPeriod] [after afterPeriod]`	Behaves the same as `no session has`, but excludes sessions via fan-out connections.
	`this session closes`	This is a shorthand form of 'no session has' that may be used to indicate that the topic is to be removed when the session that created it closes. Can have multiple `this session closes` clauses in a policy.

The meaning of the `'for'` period on `'no session has'` conditions is subtly different from on other conditions. It does not guarantee that there has been no session satisfying the condition at some point between evaluations, only that when evaluated the given period of time has passed since it was last evaluated and found to have no matching sessions.

If quotes or backslashes (\) are required within quoted values such as selectors or session criteria then they may be escaped by preceding with \.

Time period format

Time periods are specified as a number followed (with no intermediate space) by a single letter representing the time unit. The time unit may be 's' (seconds), 'm' (minutes), 'h' (hours) or 'd' (days).

For example, 10 minutes would be specified as `10m`.

Counting subscriptions and sessions

The number of subscriptions to a topic includes subscriptions that occur through routing or slave topics, as well as subscriptions to a reference topic created by a topic view (including remote topic views) from the monitored topic.

When monitoring across a cluster or using fan-out, the `'subscriptions less than'` condition is first checked on the server that owns the topic and if satisfied, each cluster member and downstream fan-out server is queried to check if the condition has also been satisfied there. The topic will only be removed if the total number of subscriptions across the cluster and at fan-out servers is less than that specified in the condition.

A `'no session has'` condition will consider all sessions hosted across a cluster and at downstream fan-out servers.

If a reference topic created by a topic view is then replicated by a remote topic view or by fan-out, it is not counted for the purposes of evaluating `'subscriptions less than'` and `'no session has'` clauses.

To avoid counting subscriptions or sessions through fan-out links, you can use the `local` keyword.

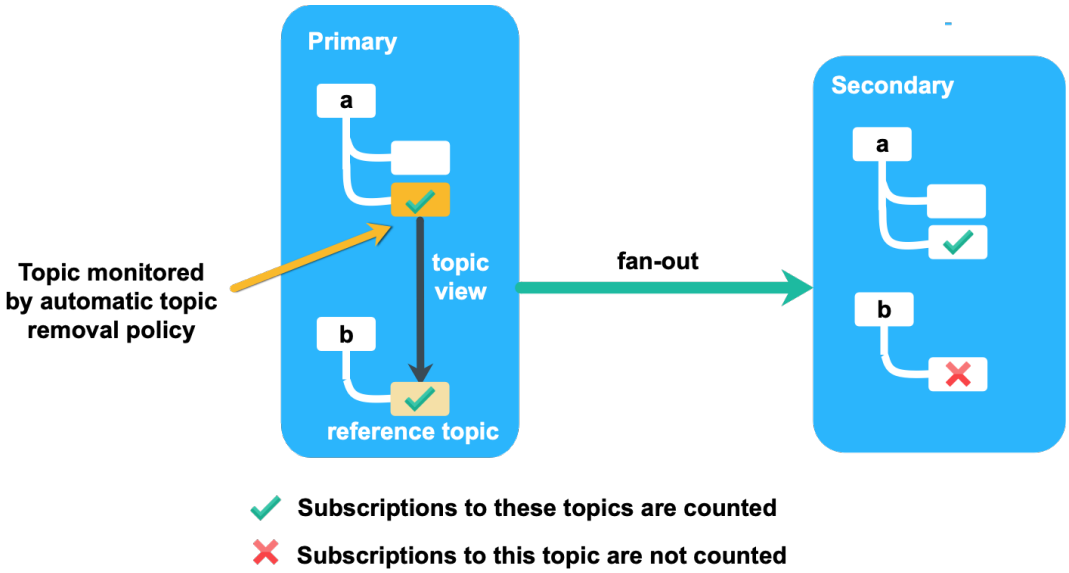


Figure 20: Automatic topic removal, topic views and fan-out

Receiving topic notifications

Receive topic notifications using topic selectors. This enables a client to receive updates when topics are added or removed, without the topic values.

Note: Topic notifications are supported by the Android API, Java API and JavaScript API.

The client must register a listener object to receive notifications about selected topics. Use a topic selector to specify the topics.

For more details about topic notifications, see Topic notifications on page 98.

Required permissions: select_topic and read_topic permissions for the specified topics

Receiving topic notifications

A client can register to receive notifications about a set of topics via a listener object.

JavaScript

```
var listener = {
    onDescendantNotification: function(topicPath, type) {},
    onTopicNotification: function(topicPath, topicSpecification,
 type) {},
    onClose: function() {},
    onError: function(error) {}
};

session.notifications.addListener(listener).then(function(reg) {
    reg.select("foo");
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
```

```
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Features.Control.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation of a listener for topic notifications.
    /// </summary>
    public sealed class TopicNotificationListener {
        private const string TOPIC_PREFIX = "topic-notifications";

        public TopicNotificationListener(string serverUrl) {
            var selector = $"?{TOPIC_PREFIX}//";

            var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open( serverUrl );

            var notifications = session.TopicNotifications;

            INotificationRegistration registration = null;
            string path = string.Empty;

            // Register a listener to receive topic notifications.
            try
            {
                registration = await
 notifications.AddListenerAsync(new Listener());
            }
            catch(Exception ex)
            {
                WriteLine($"Failed to add listener : {ex}.");
                session.Close();
                return;
            }

            // Start receiving notifications.
```

```
            try
            {
                await registration.SelectAsync(selector);
            }
            catch (Exception ex)
            {
                WriteLine($"Selector '{selector}' registration
 failed : {ex}.");
                session.Close();
                return;
            }

            // Add a topic.
            try
            {
                path = $"{TOPIC_PREFIX}/
{DateTime.Now.ToFileTimeUtc()}";
                var specification =
 session.TopicControl.NewSpecification(TopicType.STRING);

                await session.TopicControl.AddTopicAsync(path,
 specification);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add topic '{path}' : {ex}.");
                session.Close();
                return;
            }

            // Remove the topic.
            try
            {
                await session.TopicControl.RemoveTopicsAsync(path);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to remove topic '{path}' :
 {ex}.");
                session.Close();
                return;
            }

            // Stop receiving notifications.
            try
            {
                await registration.DeselectAsync(selector);
            }
            catch (Exception ex)
            {
                WriteLine($"Deselection failed for selector
 '{selector}' : {ex}.");
                session.Close();
                return;
            }

            // Unregister the listener.
            try
            {
                await registration.CloseAsync();
            }
            catch (Exception ex)
            {
```

```
                WriteLine($"Failed to unregister the listener :
 {ex}.");
            }

            // Close the session
            session.Close();
        }

        /// <summary>
        /// The listener for topic notifications.
        /// </summary>
        private class Listener : ITopicNotificationListener {
            /// <summary>
            /// Indicates that the stream was closed.
            /// </summary>
            public void OnClose()
            {
                WriteLine("The listener was closed.");
            }

            /// <summary>
            /// Notification for an immediate descendant of a
 selected topic path.
            /// </summary>
            public void OnDescendantNotification(string topicPath,
 NotificationType type)
            {
                WriteLine($"Descendant topic '{topicPath}' has been
 {type}.");
            }

            /// <summary>
            /// Indicates an error received by the callback.
            /// </summary>
            public void OnError(ErrorReason errorReason)
            {
                WriteLine($"The listener received the error:
 '{errorReason}'.");
            }

            /// <summary>
            /// Notification for a selected topic.
            /// </summary>
            public void OnTopicNotification(string topicPath,
 ITopicSpecification specification,
                                            NotificationType type)
            {
                WriteLine($"Topic '{topicPath}' has been {type}.");
            }
        }
    }
}
```

Java and Android

```
final TopicNotifications notifications =
 session.feature(TopicNotifications.class);

final TopicNotificationListener listener = new
 TopicNotificationListener() {
    @Override
```

```
    public void onTopicNotification(String topicPath,
 TopicSpecification specification, NotificationType type) {
        // Handle notifications for selected/deselected topics
    }

    @Override
    public void onDescendantNotification(String topicPath,
 NotificationType type) {
        // Handle notifications for immediate descendants
    }


    @Override
    public void onClose() {
        // The listener has been closed
    }

    @Override
    public void onError(ErrorReason error) {
        // The listener has encountered an error
    }
};

final CompletableFuture<NotificationRegistration> future =
 notifications.addListener(listener);
final NotificationRegistration registration = future.get();

registration.select("foo");
```

Updating topics

A client can use the TopicUpdate feature to update topics.

A session can update a topic in one of the following ways:

Stateless set

Stateless set is a simple way to update the value of a topic.

The client does not retain any state information about the topic, meaning that delta streaming is not possible. If a session is expected to send frequent updates to a topic, consider using an update stream instead of stateless set.

Optimistic update streams

The TopicUpdate feature supports optimistic, non-exclusive update streams.

This stream type can use delta streaming like an exclusive updater, but does not prevent other sessions from updating the topic. It can detect when it no longer knows the latest value of the topic.

In addition, you can use the `addAndSet` method, which updates a specified topic if it is present, and creates and updates it if it is not present. This can be used statelessly or using an update stream.

JSON topics also support partial updates, a separate mechanism which enables you to update part of the topic value by applying a JSON Patch.

Conditional updates

Both stateless set and optimistic update streams support conditional updates.

Conditional updates enable a client to apply a constraint to a topic update. The topic is only updated if the constraint is satisfied (as evaluated on the Diffusion server).

Constraints can check the existence of the topic, the current value of the topic, or the existence of a session lock.

You can use conditional updates to enable coordination between sessions.

If your application requires that a particular session has exclusive access to a topic, you can achieve this by making updates conditional on having a session lock on the topic.

Partial updates

JSON topics support partial updating. This is carried out with an `applyJsonPatch` method, which works separately from stateless set and optimistic updates discussed below.

The advantage of making a partial update is that a client can change part of the value of the JSON topic without needing the full value. In addition, for a large or complex JSON value, partial updating can be faster than updating the whole value.

To make a partial update, you specify a patch using the JSON Patch standard (RFC 6902).

Patches are a sequence of JSON Patch operations contained in an array. The operations are applied as an atomic update to the previous value if the resulting update is successfully calculated.

The available operations are:

- Add: `{"op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ]}`
- Remove: `{"op": "remove", "path": "/a/b/c"}`
- Replace: `{"op": "replace", "path": "/a/b/c", "value": 43}`
- Move: `{"op": "move", "from": "/a/b/c", "path": "/a/b/d"}`
- Copy: `{"op": "copy", "from": "/a/b/c", "path": "/a/b/e"}`
- Test: `{"op": "test", "path": "/a/b/c", "value": "foo"}`

The following patch will check if the value at a specific key is equal to 22, and update it to 23 if the expected value is correct:

```
[{"op":"test", "path":"/price", "value" : 22}, {"op":"add", "path":"/
price", "value": 23}]
```

The test operation checks that the CBOR representation of the value of a topic is identical to the value provided in the patch after converting it to CBOR. If the value is represented differently as CBOR, for example due to different key ordering, then the patch will return the index of the failed operation. For example, the values `{"foo": "bar", "count": 43}` and `{"count": 43, "foo": "bar"}` would not be considered equal, even though they have the same meaning.

A partial update can optionally take a constraint as with a conditional update. The patch will only be applied if the constraint is satisfied (this is evaluated prior to any JSON Patch test operations that might be specified within the patch).

Updating a topic with stateless set

Required permissions: update_topic

The `set` method replaces the current topic value with a new value.

Stateless set requires the following parameters:

Type

The type of the value.

Topic path

The path of the topic.

Value

The new value of the topic.

The primitive topic types (int64, double and string) support being set to null, but other types must be set to a value.

> **Note:** From 6.2, if a primitive topic is set to null, new subscribers are not notified of the topic value until it changes to a non-null value.

The type of the topic being updated must match the type of value it is being set to.

Updating a topic with an optimistic non-exclusive updater

Required permissions: update_topic

To create an optimistic, non-exclusive update stream, a session must specify:

Type

The type of the value.

Topic path

The path of the topic.

The update stream is created immediately without interacting with the server.

1. The stream can be updated with either `set` or `addAndSet`.
2. On the first update operation, the stream is validated with the server. From then on, the stream can detect if there are any changes to the topic it is updating.
3. If another session changes the topic, the update stream is invalidated and will stop accepting new values. Only one update stream at a time can be valid. Once a stream is invalidated, any attempt to use it results in an `InvalidUpdateStreamException`.
4. The `validate` method validates the stream with the server without setting a new value.

Related reference

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Session locks

Session locks are a way to ensure that only one session at a time can access a particular resource. For example, you can use a session lock to ensure that only one session is allowed to update a certain topic.

Session locks are a mechanism managed by the Diffusion server to coordinate access to shared resources among multiple sessions.

A session can acquire a lock, identified by a lock name (chosen by you to suit your application). Once a session acquires a lock, no other session can acquire the same lock.

Acquiring a lock does not automatically change anything else about a session. Locks are not linked to topics or permissions, except through your application's logic. It is up to you to design a suitable locking scheme and ensure your application implements it. For example, if you want to implement exclusive updating of a topic using a session lock, you must make sure that each session always acquires the lock and uses a lock constraint created from the lock when updating the topic.

By default, a lock is released when the session owning it closes. Alternatively, when acquiring a lock, a session can specify that the lock will be released if connection to the server is lost. This is done using a `scope` parameter.

A session can also explicitly release a lock.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Features.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;

namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Client implementation that demonstrates session locks.
    /// </summary>
    public sealed class SessionLocks
    {
        private ISession session1, session2;
        private ISessionLock sessionLock1, sessionLock2;

        private static string LOCK_NAME = "lockA";

        public SessionLocks(string serverUrl)
        {
            session1 =
 Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);
```

```
            session2 =
Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors) =>
CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            WriteLine("Sessions 1 and 2 have been created.");

            AcquireLockSession1();
        }

        private async void AcquireLockSession1()
        {
            try
            {
                WriteLine("Requesting lock 1...");

                sessionLock1 = await session1.LockAsync(LOCK_NAME,
SessionLockScope.UNLOCK_ON_CONNECTION_LOSS);

                WriteLine("Lock 1 has been acquired.");

                AcquireLockSession2();

                Thread.Sleep(1000);

                ReleaseLock1();
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to get lock 1 : {ex}.");
                session1.Close();
                session2.Close();
            }
        }

        private async void AcquireLockSession2()
        {
            try
            {
                WriteLine("Requesting lock 2...");

                sessionLock2 = await session2.LockAsync(LOCK_NAME,
SessionLockScope.UNLOCK_ON_CONNECTION_LOSS);

                WriteLine("Lock 2 has been acquired.");

                Thread.Sleep(1000);

                ReleaseLock2();
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to get lock 2 : {ex}.");
                session1.Close();
                session2.Close();
            }
        }

        private async void ReleaseLock1()
        {
            try
```

```
            {
                WriteLine("Requesting lock 1 release...");

                await sessionLock1.UnlockAsync();

                WriteLine("Lock 1 has been released.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to release lock 1 : {ex}.");
                session1.Close();
                session2.Close();
            }
        }

        private async void ReleaseLock2()
        {
            try
            {
                WriteLine("Requesting lock 2 release...");

                await sessionLock2.UnlockAsync();

                WriteLine("Lock 2 has been released.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to release lock 2 : {ex}.");
            }
            finally
            {
                session1.Close();
                session2.Close();
            }
        }
    }
}
```

Acquiring a lock

Required permissions: acquire_lock

Session locks are established on demand. There is no separate operation to create or destroy a named lock.

If a session attempts to acquire a lock that is not assigned, the server assigns it immediately to the session.

If a session attempts to acquire a lock that is already assigned, the server will record that the session is waiting to acquire it. When a lock is released and multiple sessions are waiting to acquire it, the server will arbitrarily assign it to one of the waiting sessions.

A session can request a lock with these parameters:

Lock name

A name for the lock.

Lock scope (optional)

The scope of the lock.

By default, the scope is `UNLOCK_ON_SESSION_LOSS`, meaning that the lock will be released when the session is closed.

If the scope is set to `UNLOCK_ON_CONNECTION_LOSS`, the lock will be released when the session loses its current connection to the server.

Using time series topics

A client can subscribe to a time series topic using a value stream, query to retrieve values within a range, append new values, or apply an edit event to override the value of an earlier event.

A time series topic stores an ordered series of events.

Each event stores a value, and has associated metadata.

An event value can be binary, double, int64, JSON, string or recordV2 value. Every event within a given time series has values with the same data type.

Table 33: Time series event metadata

Sequence number	Timestamp	Author
Unique number within time series, assigned when event created. Number increased by one with each new event.	Timestamp for event. Not guaranteed unique.	Principal that created the event. May be ANONYMOUS if session was not authorised.

Subscribing to a time series topic

Required permissions: read_topic

A client session can subscribe to a time series topic using a value stream to receive the latest events.

On subscribing to a time series topic, a session receives a set of the most recent events. By default, the latest event is sent. You can configure the TIME_SERIES_SUBSCRIPTION_RANGE property to determine how many recent events a new subscriber will receive.

Setting the DONT_RETAIN_VALUE property to true will prevent an initial event being sent, unless you have configured a subscription range.

Appending to a time series topic

Required permissions: update_topic

A session can append a value to a time series. The server will assign metadata to the event. The timestamp will be set to the current server time by default. The session can provide a timestamp, which must not be before the latest event stored by the time series topic. The author will be set to the authenticated principal of the client session. The sequence number will be one higher than the previous event in the time series.

Updating a time series topic

Required permissions: update_topic

A session can update a time series topic using the standard topic update methods. This allows for creating a topic with an initial value, using update constraints and update streams.

Editing a time series topic

Required permissions: update_topic, edit_time_series_events or edit_own_time_series_events

A client session can edit a time series. This provides a new value for an existing event.

The server retains both the original event and the edit event.

Subscribers receive two sets of metadata: the metadata for the edit event, and the metadata of the original event that was replaced.

Consider this example time series containing two events:

Sequence	Value	Type
0	A	original event
1	B	original event

Now an edit event is applied to the event with sequence number 0, changing the value to X. This information is now stored on the server:

Sequence	Value	Type
0	A	original event
1	B	original event
2	X	edit of sequence 0

The edit event is assigned a sequence number like a normal event. Both the original event 0 and the edit event are retained on the server.

If an original event has several edit events, the latest edit event (the one with the highest sequence number) determines its current value. Each edit event refers to an original event, never to another edit event.

For example, suppose another edit event is applied to change the value of the first event from X to Y. Now the information stored looks like this:

Sequence	Value	Type
0	A	original event
1	B	original event
2	X	edit of sequence 0
3	Y	edit of sequence 0

Querying a time series topic

There are two ways to query a time series topic and select a range of events, which differ only in how they handle edit events.

Value range query

Required permissions: read_topic

A value range query returns part of a time series, using the latest available value for each event.

Events are returned in order of the original sequence number. If an event has never been edited, it is simply returned. If it has been edited, the most recent edit event is returned instead.

For example, consider the example time series above after the two edits have been applied. A value range query which selected the whole topic would return:

Sequence	Value	Original event sequence
3	Y	0
1	B	-

The original value of the first event is not returned. The fact that the metadata of the original event is provided tells you that the event was edited.

A value range query is suitable for most use cases. If the client only needs the most recent value, or your application is not using edit events at all, use a value range query.

Edit range query

Required permissions: read_topic, query_obsolete_time_series_events

An edit range query can provide the history of values for events that have been edited. You are only likely to use this if you are implementing auditing or administrative features.

Because the history of edits is potentially sensitive information, an edit range query requires the additional `query_obsolete_time_series_events` permission.

There are two types of edit range query:

All edits

An all edits query returns all original events selected by the query, and all subsequent edit events that affect the originals. The results are provided in time series order.

An all edits query which selected all of the example above would return:

Sequence	Value	Original event sequence
0	A	-
1	B	-
2	X	0
3	Y	0

Both of the edit events for the original event 0 are returned.

This sort of query provides the maximum amount of information about the edit history of event values.

Latest edits

A latest edits query returns all original events selected by the query, plus the most recent edit event for each original event. The results are provided in time series order.

A latest edits query which selected all of the example above would return:

Sequence	Value	Original event sequence
0	A	N/A
1	B	N/A
3	Y	0

There were two edit events applied to the original event 0, but only the most recent edit event is returned in the query result.

This sort of query is useful if you need the original and latest value for an event, but not any intermediate values.

Note: Time series topics only retain a range of the most recent events (configured with the TIME_SERIES_RETAINED_RANGE property). By default, only the ten most recent events are retained, counting both original and edit events. Range queries only return edit events if the original event is selected. If you find that your queries do not return the results you expect, you may need to increase the retained range.

Related concepts

Time series topics on page 83
A time series topic holds a sequence of events.

Example: Publish a time series

The following example uses the Diffusion API to create and update a time series topic.

This example creates a time series topic at foo/timeseries. It demonstrates how to append and edit values.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that adds a time series topic and
 updates its value.
    /// </summary>
    public sealed class TimeSeriesPublish {

        public TimeSeriesPublish(string serverUrl) {
            string TOPIC_PREFIX = "time-series";
```

```
            var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open( serverUrl );

            // Create a time series topic with string values.
            var typeName =
 Diffusion.DataTypes.Get<string>().TypeName;
            var topic = $"{TOPIC_PREFIX}/{typeName}/
{DateTime.Now.ToFileTimeUtc()}";
            var specification =
 session.TopicControl.NewSpecification(TopicType.TIME_SERIES)

 .WithProperty(TopicSpecificationProperty.TimeSeriesEventValueType,
 typeName);

            // Add a time series topic.
            try
            {
                await session.TopicControl.AddTopicAsync(topic,
 specification);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add topic '{topic}' : {ex}.");
                session.Close();
                return;
            }

            // Append a value to the time series topic using a custom
 timestamp.
            try
            {
                await session.TimeSeries.AppendAsync<string>(topic,
 "Value1",
                    DateTimeOffset.FromUnixTimeMilliseconds(322));
            }
            catch (Exception ex)
            {
                WriteLine($"Topic {topic} value could not be
 appended : {ex}.");
                session.Close();
                return;
            }

            // Append a value to the time series topic. The timestamp
 will be set to the current server time.
            try
            {
                await session.TimeSeries.AppendAsync<string>(topic,
 "Value 1");
                await session.TimeSeries.AppendAsync<string>(topic,
 "Value 2");
                await session.TimeSeries.AppendAsync<string>(topic,
 "Value 3");
                await session.TimeSeries.AppendAsync<string>(topic,
 "Value 4");
            }
            catch (Exception ex)
            {
                WriteLine($"Topic {topic} value could not be
 appended : {ex}.");
```

```
                session.Close();
                return;
            }

            // Edit a value of the time series topic.
            try
            {
                // Edits the time series topic with sequence number 1
 and value 'Value1'.
                await session.TimeSeries.EditAsync<string>(topic, 1,
 "Value 1a");
            }
            catch (Exception ex)
            {
                WriteLine($"Topic {topic} value could not be edited :
 {ex}.");
                session.Close();
                return;
            }

            // Update the time series topic using the standard topic
 update method.
            try
            {
                var newValue = "Last Value";

                await session.TopicUpdate.SetAsync(topic, newValue);
            }
            catch (Exception ex)
            {
                WriteLine($"Topic {topic} could not be updated :
 {ex}.");
                session.Close();
                return;
            }

            // Remove the topic.
            try
            {
                await
 session.TopicControl.RemoveTopicsAsync( topic );
            }
            catch(Exception ex)
            {
                WriteLine( $"Failed to remove topic '{topic}' :
 {ex}." );
            }

            // Close the session.
            session.Close();
        }
    }
}
```

Java and Android

```
/
*******************************************************************************
 * Copyright (C) 2017, 2020 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
```

```
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 ********************************************************************************
package com.pushtechnology.diffusion.examples;

import static
 com.pushtechnology.diffusion.client.Diffusion.newTopicSpecification;
import static
 com.pushtechnology.diffusion.datatype.DataTypes.INT64_DATATYPE_NAME;

import java.time.Instant;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.features.TimeSeries;
import
 com.pushtechnology.diffusion.client.features.TimeSeries.EventMetadata;
import com.pushtechnology.diffusion.client.features.TopicUpdate;
import
 com.pushtechnology.diffusion.client.features.control.topics.TopicControl;
import com.pushtechnology.diffusion.client.session.Session;
import
 com.pushtechnology.diffusion.client.topics.details.TopicSpecification;
import com.pushtechnology.diffusion.client.topics.details.TopicType;

/**
 * This example shows a control client creating a {@link TimeSeries}
 topic.
 * Values can be appended to the topic using {@link
 #appendValue(long)}, and
 * the last value of the topic can be edited using {@link
 #editLast(long)}.
 * Alternatively, the methods provided by the {@link TopicUpdate}
 feature can be
 * used. See {@link TopicUpdateExample} for example usages of this
 API.
 *
 * @author Push Technology Limited
 * @since 6.0
 * @see ClientConsumingTimeSeriesTopics
 * @see TimeSeriesQueryExample
 */
public class ControlClientUpdatingTimeSeriesTopics {

    private static final String TOPIC_PATH = "foo/timeseries";
    private static final Logger LOG =

 LoggerFactory.getLogger(ControlClientUpdatingTimeSeriesTopics.class);
```

```
    private final Session session;
    private final TimeSeries timeSeries;
    private final TopicControl topicControl;

    /**
     * Constructor.
     *
     * @param serverUrl server URL to connect to example "ws://
diffusion.example.com:80"
     */
    public ControlClientUpdatingTimeSeriesTopics(String serverUrl)
        throws InterruptedException, ExecutionException,
 TimeoutException {

        session =
 Diffusion.sessions().principal("control").password("password")
            .open(serverUrl);

        timeSeries = session.feature(TimeSeries.class);
        topicControl = session.feature(TopicControl.class);

        final TopicSpecification spec =
 newTopicSpecification(TopicType.TIME_SERIES)

 .withProperty(TopicSpecification.TIME_SERIES_EVENT_VALUE_TYPE,
 INT64_DATATYPE_NAME);

        topicControl.addTopic(TOPIC_PATH, spec)
            .thenAccept(result -> LOG.info("Add topic result: {}",
 result)).get(5, TimeUnit.SECONDS);
    }

    /**
     * Appends a value to the time series topic.
     *
     * @param value value to append
     * @return the event metadata from the successful append
     */
    public EventMetadata appendValue(long value)
        throws IllegalArgumentException, InterruptedException,
 ExecutionException, TimeoutException {
        return timeSeries.append(TOPIC_PATH, Long.class,
 value).get(5, TimeUnit.SECONDS);
    }

    /**
     * Close the session and remove the time series topic.
     */
    public void close()
        throws IllegalArgumentException, InterruptedException,
 ExecutionException, TimeoutException {
        topicControl.removeTopics("?foo//").get(5, TimeUnit.SECONDS);
        session.close();
    }

    /**
     * Edit the last value in a time series topic.
     *
     * @param value value to edit with
     */
    public void editLast(long value) {
        //Obtain the last value in the time series topic
```

```
timeSeries.rangeQuery().fromLast(1).as(Long.class).selectFrom(TOPIC_PATH)
            .whenComplete((query, ex) -> {
                if (ex != null) {
                    LOG.error("Error obtaining the range query: {}",
ex);
                    return;
                }
                //Perform the value edit
                query.stream().forEach(event -> {
                    timeSeries.edit(TOPIC_PATH, event.sequence(),
Long.class, value)
                        .whenComplete((metadata, e) -> {
                            if (e != null) {
                                LOG.error("Error editing topic: {}",
e);
                                return;
                            }
                            LOG.info("EventMetadata from edit: {}",
metadata);
                        });
                });
            });
    }

    /**
     * Appends a value to the time series topic. Allows for creation
of events
     * with a custom timestamp. This can be used for loading historic
or future
     * values.
     *
     * @param value value to append
     * @param timestamp the user supplied timestamp
     * @return the event metadata from the successful append
     */
    public EventMetadata appendValue(long value, Instant timestamp)
        throws IllegalArgumentException, InterruptedException,
ExecutionException, TimeoutException {
        return timeSeries.append(TOPIC_PATH, Long.class, value,
timestamp).get(5, TimeUnit.SECONDS);
    }
}
```

Example: Subscribe to a time series

The following example uses Diffusion API to subscribe to a time series topic.

This example demonstrates subscribing to a time series topic at foo/timeseries.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
```

```
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Features.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;
using System;
using PushTechnology.ClientInterface.Client.Features.TimeSeries;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation which subscribes to a string time
 series topic and
    /// consumes the data it receives.
    /// </summary>
    public sealed class TimeSeriesTopics {

        public TimeSeriesTopics(string serverUrl) {
            string TOPIC_PREFIX = "time-series";

            var session =
 Diffusion.Sessions.Principal("client").Password("password").Open(serverUrl);

            var typeName =
 Diffusion.DataTypes.Get<String>().TypeName;
            var topic = $"?{TOPIC_PREFIX}/{typeName}//";

            // Add a value stream
            var stringStream = new StringStream();
            session.Topics.AddTimeSeriesStream<string>(topic,
 stringStream);

            // Subscribe to the topic.
            try
            {
                await session.Topics.SubscribeAsync(topic);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to subscribe to topic '{topic}' :
 {ex}.");
            }
            finally
            {
                // Note that closing the session, will automatically
 unsubscribe from all topics
                // the client is subscribed to.
                session.Topics.RemoveStream(stringStream);
                session.Close();
            }
        }
```

```
        /// <summary>
        /// Basic implementation of the IValueStream for time series
 string topics.
        /// </summary>
        private sealed class StringStream :
 IValueStream<IEvent<string>> {
            /// <summary>
            /// Notification of the stream being closed normally.
            /// </summary>
            public void OnClose()
                => WriteLine( "The subscription stream is now
 closed." );

            /// <summary>
            /// Notification of a contextual error related to this
 callback.
            /// </summary>
            /// <param name="errorReason">Error reason.</param>
            public void OnError( ErrorReason errorReason )
                => WriteLine( $"An error has occured :
 {errorReason}." );

            /// <summary>
            /// Notification of a successful subscription.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            public void OnSubscription( string topicPath,
 ITopicSpecification specification )
                => WriteLine( $"Client subscribed to {topicPath}." );

            /// <summary>
            /// Notification of a successful unsubscription.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            /// <param name="reason">Error reason.</param>
            public void OnUnsubscription( string topicPath,
 ITopicSpecification specification,
                                          TopicUnsubscribeReason
 reason )
                => WriteLine( $"Client unsubscribed from
 {topicPath} : {reason}." );

            /// <summary>
            /// Topic update received.
            /// </summary>
            /// <param name="topicPath">Topic path.</param>
            /// <param name="specification">Topic specification.</
param>
            /// <param name="oldValue">Value prior to update.</param>
            /// <param name="newValue">Value after update.</param>
            public void OnValue( string topicPath,
 ITopicSpecification specification,
                                 IEvent<string> oldValue,
 IEvent<string> newValue )
                => WriteLine( $"New value of {topicPath} is
 {newValue}." );
        }
    }
```

```
}
```

Java and Android

```
/
*******************************************************************************
 * Copyright (C) 2017 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 *******************************************************************************
package com.pushtechnology.diffusion.examples;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.features.TimeSeries;
import com.pushtechnology.diffusion.client.features.TimeSeries.Event;
import com.pushtechnology.diffusion.client.features.Topics;
import
 com.pushtechnology.diffusion.client.features.Topics.ValueStream;
import com.pushtechnology.diffusion.client.session.Session;

/**
 * This demonstrates a client session subscribing to a
 * {@link TimeSeries} topic.
 *
 * @author Push Technology Limited
 * @since 6.0
 * @see ControlClientUpdatingTimeSeriesTopics
 * @see TimeSeriesQueryExample
 */
public class ClientConsumingTimeSeriesTopics {

    private static final String TOPIC_PATH = "foo/timeseries";

    private Session session;

    /**
     * Constructor.
     *
     * @param serverUrl for example "ws://diffusion.example.com:80"
     * @param valueStream value stream to receive time series topic
 events
     */
    public ClientConsumingTimeSeriesTopics(String serverUrl,
 ValueStream<Event<Long>> valueStream)
        throws InterruptedException, ExecutionException,
 TimeoutException {
```

```
        session =
 Diffusion.sessions().principal("client").password("password")
           .open(serverUrl);

        final Topics topics = session.feature(Topics.class);
        topics.addTimeSeriesStream(TOPIC_PATH, Long.class,
 valueStream);
        topics.subscribe(TOPIC_PATH).get(5, TimeUnit.SECONDS);
    }

    /**
     * Close the session.
     */
    public void close() {
        session.close();
    }
}
```

Managing subscriptions

A client can use the SubscriptionControl feature to subscribe other client sessions to topics that they have not requested subscription to themselves and also to unsubscribe clients from topics. It also enables the client to register as the handler for routing topic subscriptions.

Subscribing and unsubscribing clients

Required permissions: modify_session, select_topic permission for the topics being subscribed to

A client can subscribe client sessions that it knows about to topics that those clients have not explicitly requested. It can also unsubscribe clients from topics.

A session identifier is required to specify the client session that is to be subscribed or unsubscribed. Use the ClientControl feature to get the identifiers for connected client sessions.

The SubscriptionControl feature uses topic selectors to specify topics for subscription and unsubscription. Many topics can be specified in a single operation.

The client being subscribed to topics must have read_topic permission for the topics it is being subscribed to.

Using session properties to select clients to subscribe and unsubscribe

Required permissions: view_session, modify_session, select_topic permission for the topics being subscribed to

When managing client subscriptions, a client can specify a filter for which client sessions it subscribes to topics or unsubscribes from topics. The filter is a query expression on the values of session properties.

The managing client defines a filter and sends a subscription request with this filter to the Diffusion server. The Diffusion server evaluates the query and subscribes those currently connected client sessions whose session properties match the filter to the topic or topics.

The filter is evaluated only once. Clients that subsequently connect or clients whose properties change are do not cause the subscription request to be reevaluated. Even if these clients match the filter, they are not subscribed.

Managing all subscriptions from a separate control session

You can prevent client sessions from subscribing themselves to topics and control all subscriptions from a separate control client session that uses SubscriptionControl feature to subscribe clients to topics.

To restrict subscription capability to control sessions, configure the following permissions:

Control session:

- Grant the modify_session permission
- Grant the select_topic permission

  This can either be granted for the default path scope or more selectively to restrict the topic selectors the control session can use.

Other sessions:

- Grant read_topic to the appropriate topics.
- Deny the select_topic permission by default.

  Do not assign the session a role that has the select_topic permission for the default path scope. This prevents the session from subscribing to all topics using a wildcard selector.
- Optionally, grant the select_topic permission to specific branches of the topic tree to which the session can subscribe freely.

Acting as a routing subscription handler

Required permissions: view_session, modify_session, register_handler

Routing topics can be created with a server-side handler that assigns clients to real topics. However, you can omit the server-side handler such that subscriptions to routing topics are directed at a client acting as a routing subscription handler.

A client can register a routing subscription handler for a branch of the topic tree. Any subscription requests to routing topics in that branch that do not have server-side handlers are passed to the client for action.

On receipt of a routing subscription request the client can respond with a route request that specifies the path of the actual topic that the routing topic maps to for the requesting client. This subscription fails if the target topic does not already exist or if the requesting client does not have read_topic permission for the routing topic or target topic.

The client can complete other actions before calling back to route. For example, it could use the TopicControl feature to create the topic that the client is to map to.

Alternatively, the client can defer the routing subscription request in which case the requesting client remains unsubscribed. This is similar to denying it from an authorization point of view.

The client must reply with a route or defer for all routing requests.

Related concepts

Topic selectors on page 44
A topic selector defines a set of topics paths that identify topics. You can create a topic selector from a topic selector expression.

Session properties on page 220
A client session has a number of properties associated with it. Properties are key-value pairs. Both the key and the value are case sensitive.

Session filtering on page 222

Session filters enable you to query the set of connected client sessions on the Diffusion server based on their session properties.

Example: Subscribe other clients to topics

The following examples use the SubscriptionControl feature in the Diffusion API to subscribe other client sessions to topics.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features.Control.Clients;
using PushTechnology.ClientInterface.Client.Features.Control.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that subscribes another session to
 topics.
    /// </summary>
    public sealed class SubscriptionControlSubscribe
    {
        public SubscriptionControlSubscribe(string serverUrl)
        {
            var topic = $"topic/example";

            var controlSession =
 Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            var subscriptionControl =
 controlSession.SubscriptionControl;

            try
            {
```

```
                await
controlSession.TopicControl.AddTopicAsync(topic, TopicType.STRING,
cancellationToken);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add topic '{topic}' : {ex}.");
                controlSession.Close();
                return;
            }

            ISession session = Diffusion.Sessions.Principal("client")

.Credentials(Diffusion.Credentials.Password("password"))
                .CertificateValidation((cert, chain, errors) =>
CertificateValidationResult.ACCEPT)
                .NoReconnection()
                .Open(serverUrl);

            WriteLine($"Session with id '{session.SessionId}'
created.");

            var subscriptionCallback = new SubscriptionCallback();

            var topicSelector = "?topic//";

            try
            {
                subscriptionControl.Subscribe(session.SessionId,
topicSelector, subscriptionCallback);

                WriteLine($"Session with id '{session.SessionId}' is
subscribed to '{topicSelector}'.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to subscribe to
'{topicSelector}' : {ex}.");
                session.Close();
                controlSession.Close();
                return;
            }

            try
            {
                subscriptionControl.Unsubscribe(session.SessionId,
topicSelector, subscriptionCallback);

                WriteLine($"Session with id '{session.SessionId}' is
unsubscribed to '{topicSelector}'.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to unsubscribe to
'{topicSelector}' : {ex}.");
            }

            // Close the sessions
            session.Close();
            controlSession.Close();
        }

        /// <summary>
```

```
        /// The callback for subscription operations.
        /// </summary>
        private class SubscriptionCallback : ISubscriptionCallback
        {
            /// <summary>
            /// Indicates that the session was closed.
            /// </summary>
            public void OnDiscard()
            {
                WriteLine("The session was closed.");
            }

            /// <summary>
            /// Indicates that a requested operation has been handled
 by the server.
            /// </summary>
            public void OnComplete()
            {
                WriteLine("Subscription complete.");
            }
        }
    }
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import java.util.concurrent.CompletableFuture;

import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.client.features.control.topics.SubscriptionControl
import com.pushtechnology.diffusion.client.session.Session;
import com.pushtechnology.diffusion.client.session.SessionId;

/**
 * This demonstrates using a client to subscribe and unsubscribe
 other clients
 * to topics.
 * <P>
 * This uses the 'SubscriptionControl' feature.
 *
 * @author Push Technology Limited
 * @since 5.0
 */
public class ControlClientSubscriptionControl {

    private final Session session;

    private final SubscriptionControl subscriptionControl;

    /**
     * Constructor.
     */
    public ControlClientSubscriptionControl() {

        session =

 Diffusion.sessions().principal("control").password("password")
                .open("ws://diffusion.example.com:80");
```

```
        subscriptionControl =
 session.feature(SubscriptionControl.class);
    }

    /**
     * Subscribe a client to topics.
     *
     * @param sessionId client to subscribe
     * @param topicSelector topic selector expression
     * @param callback for subscription result
     */
    public CompletableFuture<?> subscribe(
        SessionId sessionId,
        String topicSelector) {

        // To subscribe a client to a topic, this client session
        // must have the 'modify_session' permission.
        return subscriptionControl.subscribe(
            sessionId,
            topicSelector);
    }

    /**
     * Unsubscribe a client from topics.
     *
     * @param sessionId client to unsubscribe
     * @param topicSelector topic selector expression
     * @return a CompletableFuture that completes when a response is
 received
     *         from the server
     */
    public CompletableFuture<?> unsubscribe(
        SessionId sessionId,
        String topicSelector) {

        // To unsubscribe a client from a topic, this client session
        // must have the 'modify_session' permission.
        return subscriptionControl.unsubscribe(
            sessionId,
            topicSelector);
    }

    /**
     * Close the session.
     */
    public void close() {
        session.close();
    }
}
```

C

```
/*
 * This example waits to be notified of a client connection, and then
 * subscribes that client to a named topic.
 */

#include <stdio.h>
#include <unistd.h>
```

```
#include <apr.h>
#include <apr_thread_mutex.h>
#include <apr_thread_cond.h>

#include "diffusion.h"
#include "args.h"

ARG_OPTS_T arg_opts[] = {
        ARG_OPTS_HELP,
        {'u', "url", "Diffusion server URL", ARG_OPTIONAL,
 ARG_HAS_VALUE, "ws://localhost:8080"},
        {'p', "principal", "Principal (username) for the connection",
 ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'c', "credentials", "Credentials (password) for the
 connection", ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'t', "topic_selector", "Topic selector to subscribe/
unsubscribe clients from", ARG_OPTIONAL, ARG_HAS_VALUE, ">foo"},
        END_OF_ARG_OPTS
};
HASH_T *options = NULL;

/*
 * Callback invoked when a client has been successfully subscribed to
 * a topic.
 */
static int
on_subscription_complete(SESSION_T *session, void *context)
{
        printf("Subscription complete\n");
        return HANDLER_SUCCESS;
}

/*
 * Callback invoked when a client session has been opened.
 */
static int
on_session_open(SESSION_T *session, const SESSION_PROPERTIES_EVENT_T
 *request, void *context)
{
        if(session_id_cmp(*session->id, request->session_id) == 0) {
                // It's our own session, ignore.
                return HANDLER_SUCCESS;
        }

        char *topic_selector = hash_get(options, "topic_selector");

        char *sid_str = session_id_to_string(&request->session_id);
        printf("Subscribing session %s to topic selector %s\n",
 sid_str, topic_selector);
        free(sid_str);

        /*
         * Subscribe the client session to the topic.
         */
        SUBSCRIPTION_CONTROL_PARAMS_T subscribe_params = {
                .session_id = request->session_id,
                .topic_selector = topic_selector,
                .on_complete = on_subscription_complete
        };
        subscribe_client(session, subscribe_params);

        return HANDLER_SUCCESS;
```

```
}

int
main(int argc, char **argv)
{
        /*
         * Standard command-line parsing.
         */
        options = parse_cmdline(argc, argv, arg_opts);
        if(options == NULL || hash_get(options, "help") != NULL) {
                show_usage(argc, argv, arg_opts);
                return EXIT_FAILURE;
        }

        const char *url = hash_get(options, "url");
        const char *principal = hash_get(options, "principal");
        CREDENTIALS_T *credentials = NULL;
        const char *password = hash_get(options, "credentials");
        if(password != NULL) {
                credentials = credentials_create_password(password);
        }

        /*
         * Create a session with Diffusion.
         */
        DIFFUSION_ERROR_T error = { 0 };
        SESSION_T *session = session_create(url, principal,
 credentials, NULL, NULL, &error);
        if(session == NULL) {
                fprintf(stderr, "Failed to create session: %s\n",
 error.message);
                return EXIT_FAILURE;
        }

        /*
         * Register a session properties listener, so we are notified
         * of new client connections.
         * In the callback, we will subscribe the client to topics
         * according to the topic_selector argument.
         */
        SET_T *required_properties = set_new_string(1);
        set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_FIXED_PROPERTIES);
        SESSION_PROPERTIES_REGISTRATION_PARAMS_T params = {
                .on_session_open = on_session_open,
                .required_properties = required_properties
        };
        session_properties_listener_register(session, params);
        set_free(required_properties);

        /*
         * Pretend to do some work.
         */
        sleep(10);

        /*
         * Close session and tidy up.
         */
        session_close(session, NULL);
        session_free(session);

        return EXIT_SUCCESS;
}
```

Python

```
#  Copyright (c) 2021 Push Technology Ltd., All Rights Reserved.
#
#  Use is subject to license terms.
#
#  NOTICE: All information contained herein is, and remains the
#  property of Push Technology. The intellectual and technical
#  concepts contained herein are proprietary to Push Technology and
#  may be covered by U.S. and Foreign Patents, patents in process,
 and
#  are protected by trade secret or copyright law.
""" Example of sending a request to a session filter. """

import asyncio
import diffusion
import diffusion.features.topics as topics

# Diffusion server connection information; same for both sessions
# adjust as needed for the server used in practice
server_url = "ws://localhost:8080"
principal = "admin"
credentials = diffusion.Credentials("password")


# stream callback functions
def on_update(*, old_value, topic_path, topic_value, **kwargs):
    print("Topic:", topic_path)
    if old_value is None:
        print("  Initial value:", topic_value)
    else:
        print("  Value updated")
        print("    Old value:", old_value)
        print("    New value:", topic_value)


def on_subscribe(*, topic_path, **kwargs):
    print(f"Subscribed to {topic_path}")


def on_unsubscribe(*, reason, topic_path, **kwargs):
    print(f"Unsubscribed from {topic_path} because {str(reason)}")


# example properties
topic_selector = "foo/bar"
topic_type = diffusion.datatypes.STRING

session_duration = 15

# value stream instance
value_stream = topics.ValueStreamHandler(
    data_type=topic_type,
    update=on_update,
    subscribe=on_subscribe,
    unsubscribe=on_unsubscribe,
)


# Because Python SDK for Diffusion is async, all the code needs to be
```

```
# wrapped inside a coroutine function, and executed using
 asyncio.run.
async def main():

    # creating the session
    async with diffusion.Session(
        url=server_url, principal=principal, credentials=credentials
    ) as session:

        # PUSH_SNIPPET_BEGIN: [topic_notifications.xml/subscribe]
        print(f"Adding value stream {topic_selector}")
        session.topics.add_value_stream(
            topic_selector=topic_selector, stream=value_stream
        )
        print(f"Subscribing to {topic_selector}")
        await session.topics.subscribe(topic_selector)
        # PUSH_SNIPPET_END: [topic_notifications.xml/subscribe]
        await asyncio.sleep(session_duration)

        print(f"Unsubscribing from {topic_selector}")
        await session.topics.unsubscribe(topic_selector)

        await asyncio.sleep(5)  # keep alive to display the
 unsubscription message


if __name__ == "__main__":
    asyncio.run(main())
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Example: Receive notifications when a client subscribes to a routing topic

The following examples use the SubscriptionControl feature in the Diffusion API to listen for notifications of when a client subscribes to a routing topic.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using System.Threading.Tasks;
```

```
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Features.Control.Clients;
using PushTechnology.ClientInterface.Client.Features.Control.Topics;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Topics;
using PushTechnology.ClientInterface.Client.Topics.Details;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that routes subscriptions.
    /// </summary>
    public sealed class SubscriptionControlRoutingHandler
    {
        public SubscriptionControlRoutingHandler(string serverUrl)
        {
            var topic = "branch/routingTopic";

            var controlSession =
 Diffusion.Sessions.Principal("control").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            var subscriptionControl =
 controlSession.SubscriptionControl;

            try
            {
                await controlSession.TopicControl.AddTopicAsync(
                    topic,
 controlSession.TopicControl.NewSpecification(TopicType.ROUTING),
                    cancellationToken);

                await controlSession.TopicControl.AddTopicAsync(
                    "branch/someTopic", TopicType.STRING,
 cancellationToken);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add topic '{topic}' : {ex}.");
                controlSession.Close();
                return;
            }

            try
            {
                // Sets up a handler so that all subscriptions to
 topic 'branch' are routed.
                subscriptionControl.AddRoutingSubscriptionHandler(
                    "branch", new
 RoutingSubscriptionRequestHandler());

                WriteLine($"Routing handler added for topic
 'branch'.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to add routing handler for topic
 'branch' : {ex}.");
            }
```

```
            ISession session = Diffusion.Sessions.Principal("client")

 .Credentials(Diffusion.Credentials.Password("password"))
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .NoReconnection()
                .Open(serverUrl);

            WriteLine($"Session with id '{session.SessionId}'
 created.");

            try
            {
                await session.Topics.SubscribeAsync("?branch/",
 cancellationToken);

                WriteLine($"Session with id '{session.SessionId}' is
 subscribed to '?branch/'.");

                await Task.Delay(TimeSpan.FromMilliseconds(300));
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to subscribe to '?branch/' :
 {ex}.");
                session.Close();
                controlSession.Close();
                return;
            }

            try
            {
                WriteLine($"Removing topics for path '?branch/'.");

                var result = await
 session.TopicControl.RemoveTopicsAsync("?branch/",
 cancellationToken);

                WriteLine($"{result.RemovedCount} topics successfully
 removed.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to remove topics for path '?
branch/' : {ex}.");
            }

            // Close the sessions
            controlSession.Close();
        }

        /// <summary>
        /// The handler for routing subscription requests.
        /// </summary>
        private class RoutingSubscriptionRequestHandler :
 IRoutingSubscriptionRequestHandler
        {
            /// <summary>
            /// Called when the handler has been registered at the
 server and is now active.
            /// </summary>
            /// <param name="topicPath">Path to topic.</param>
```

```
            /// <param name="registeredHandler">Reference to a
registered handler.</param>
            public void OnActive(string topicPath, IRegisteredHandler
registeredHandler)
            {
                WriteLine($"Handler registered for '{topicPath}'.");
            }

            /// <summary>
            /// Called if the handler is closed.
            /// </summary>
            /// <param name="topicPath">Path to topic.</param>
            public void OnClose(string topicPath)
            {
                WriteLine($"Handler closed for '{topicPath}'.");
            }

            /// <summary>
            /// A request to subscribe to a specific routing topic.
            /// </summary>
            /// <param name="request">The request to subscribe to a
routing topic.</param>
            public void
OnSubscriptionRequest(IRoutingSubscriptionRequest request)
            {
                var topic = "branch/someTopic";

                try
                {
                    WriteLine($"Routing subscription to '{topic}'.");
                    request.Route(topic, new SubscriptionCallback());
                }
                catch(Exception ex)
                {
                    WriteLine($"Subscription routing failed: {ex}.");
                }
            }
        }

        /// <summary>
        /// The callback for subscription operations.
        /// </summary>
        private class SubscriptionCallback : ISubscriptionCallback
        {
            /// <summary>
            /// Indicates that the session was closed.
            /// </summary>
            public void OnDiscard()
            {
                WriteLine("The session was closed.");
            }

            /// <summary>
            /// Indicates that a requested operation has been handled
by the server.
            /// </summary>
            public void OnComplete()
            {
                WriteLine("Subscription complete.");
            }
        }
    }
```

```
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.client.features.control.topics.SubscriptionControl
import
 com.pushtechnology.diffusion.client.features.control.topics.SubscriptionControl
import
 com.pushtechnology.diffusion.client.features.control.topics.SubscriptionControl
import com.pushtechnology.diffusion.client.session.Session;

/**
 * This demonstrates using a control client to be notified of
 subscription
 * requests to routing topics.
 * <P>
 * This uses the 'SubscriptionControl' feature.
 *
 * @author Push Technology Limited
 * @since 5.0
 */
public class ControlClientSubscriptionControlRouting {

    private static final Logger LOG =

 LoggerFactory.getLogger(ControlClientSubscriptionControlRouting.class);

    private final Session session;

    /**
     * Constructor.
     */
    public ControlClientSubscriptionControlRouting(String serverUrl)
 {

        session =

 Diffusion.sessions().principal("control").password("password")
                .open(serverUrl);

        final SubscriptionControl subscriptionControl =
            session.feature(SubscriptionControl.class);

        // Sets up a handler so that all subscriptions to topic a/b
 are routed
        // to routing/target/topic
        // To do this, the client session requires the
 'view_session',
        // 'modify_session', and 'register_handler' permissions.
        subscriptionControl.addRoutingSubscriptionHandler(
            "a/b",
            new RoutingHandler.Default() {
                @Override
                public void onSubscriptionRequest(
```

```
                    final RoutingSubscriptionRequest request) {

                    request.route("routing/target/
topic").whenComplete((voidResult, exception) -> {
                            if (exception != null) {
                                LOG.info("subscription routing
 failed", exception);
                            }
                    });
                }
            });
    }

    /**
     * Close the session.
     */
    public void close() {
        session.close();
    }
}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Using request-response messaging

You can send request messages directly to a client session, a set of client sessions, or a message path. The recipient of a message can respond to the request.

Typed requests and responses

Each request and response messages has a data type. The data type can be one of the following types:

- JSON
- Binary
- String
- Int64
- Double

The data type of the response is not required to be the same as the data type of the request it responds to.

When you send a request, you specify the data type of the request message and the data type of the response message it expects. When you register a handler or a stream to receive requests and respond to them, you specify the data type of the requests it receives and the data type of the responses it sends.

Data types are organized in a hierarchy of compatibility.

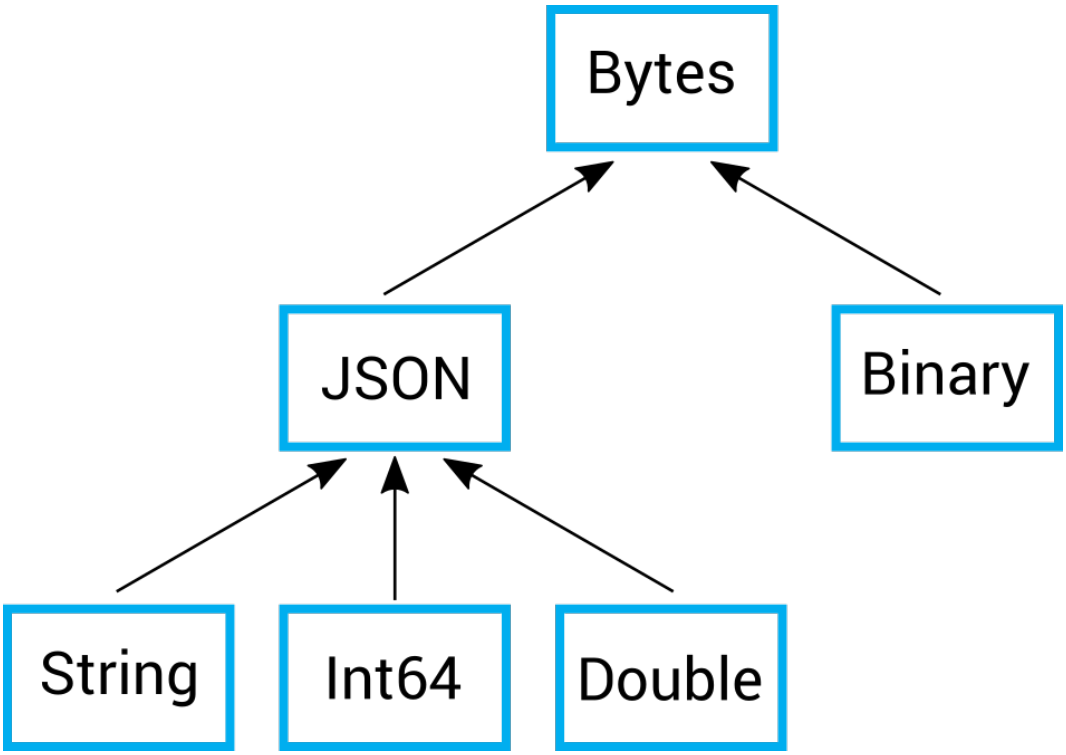


Figure 21: Data type hierarchy

A request or response with a data type at a lower (more specific) level of the hierarchy can be received by a stream, handler, or requester that is expecting a message with a data type at a higher (more general) level of the hierarchy. For example, a request message with a string data type, can be received by a stream or handler that specifies string, JSON, or bytes as the request type.

Message path

The message path is made up of path segments separated by the slash character (/). Each path segment can be made up of one or more Unicode characters. The slash character (/) is not permitted in any path segment. The restrictions for message paths are the same as those for paths at which topics can be bound. For more information, see Topic naming on page 43.

However, messaging is entirely separate from streaming data through topics:

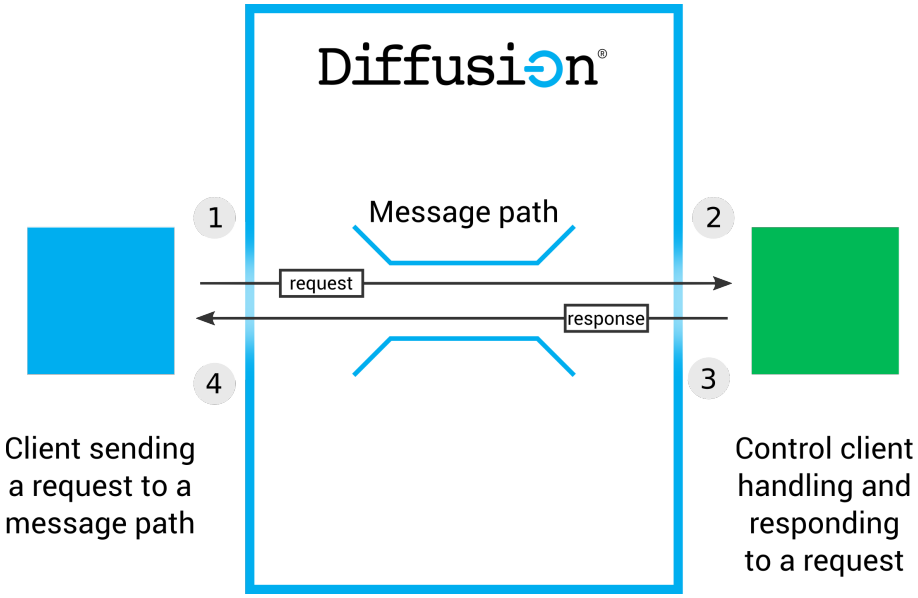
- Message paths are unrelated topic paths. Sending a message does not change the state of any topic and does not publish the message to topic subscribers.
- An application can bind a topic to a topic path and use the same path as a message path. This is a useful convention where the messages are related to the topic in some way. The messages sent to the message path do not interact with the topic in any way.
- If a topic is bound to the path used by messaging, the data type of the topic does not affect the data type of any messages sent using the message path.
- The security permissions required to use a path for messaging are separate from those required to use a topic bound to that path to stream data.

Request-response and clusters

Request-response messaging is cluster-aware. This means that a message handler attached to one server in a Diffusion cluster can route a message to any server within that cluster.

Sending request messages to a message path

A client session can send a request message containing typed data to a message path. One or more client sessions can register to handle messages sent to that message path. The handling client session can then send a response message containing typed data. The response message is sent to the requesting client session directly, through the same message path.



When a request message is sent to a message path and a client session that handlers that message path responds, the following events occur:

1. A client session sends a request message to a message path.
2. The control client session receives the request message through a request handler.
3. The session client session uses sends a response to the request message.
4. The client session receives the response.

Both the request message and the response message contain typed values. The messages can contain data of one of the following types: JSON, binary, string, 64-bit integer, or double. The response message is not required to be the same data type as the request it responds to.

Sending to a message path

Required permissions: send_to_handler permission for the specified message path

Send the request message specifying the following information:

- The message path to send the request to and receive the response through
- The request message
- The datatype of the request message
- The datatype of the response message

JavaScript

```
// Example with json topic type.
var jsonType = diffusion.datatypes.json();
// Create a JSON object to send as a request.
var requestJson = jsonType.from("hello");
```

```
// Send the request to a message path "foo".
session.messages.sendRequest('foo', requestJson,
 jsonType).then(function(response) {
console.log(response.get());
}, function(error) {});
```

.NET

```
/// <summary>
/// Client implementation that sends request messages to a path and
/// displays the response.
/// </summary>
public sealed class SendingPathRequestMessages {

    public SendingPathRequestMessages(string serverUrl) {
        string serverUrl = args[ 0 ];

        var session =
 Diffusion.Sessions.Principal("control").Password( "password" )
            .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
            .Open(serverUrl);

        var messaging = session.Messaging;
        string messagingPath = ">random/requestResponse";

        try {
            string response = await
 messaging.SendRequestAsync<string, string>(
                messagingPath, "Starting chat..." );
            WriteLine( $"Received response: '{response}'." );

            Thread.Sleep( 1000 );

            response = await messaging.SendRequestAsync<string,
 string>(
                messagingPath, "Hello!" );
            WriteLine( $"Received response: '{response}'." );
        }
        catch ( Exception e )
        {
            WriteLine( $"Got exception: '{e.Message}'." );
        }

        // Close the session
        session.Close();
    }
}
```

Java and Android

```

//Establish client sesssion
final Session session =
 Diffusion.sessions().principal("client").password("password").open("ws://
localhost:8080");

//Obtain the Messaging feature
final Messaging messaging = session.feature(Messaging.class);

//Create a JSON object to send as a request
```

```
final JSON request =
 Diffusion.dataTypes().json().fromJsonString("\"hello\"");

//Send the request to a message path "foo" and wait for (at most) 5
 seconds until the response is received.
final JSON response = messaging.sendRequest("foo", request,
 JSON.class, JSON.class).get(5, TimeUnit.SECONDS)
```

Python

```
# Sending the request and receiving the response.
print(f"Sending request: '{request}' to path '{path}'...")
try:
    response = await session.messaging.send_request_to_path(
        path=path, request=request_type(request)
    )
except diffusion.DiffusionError as ex:
    print(f"ERROR: {ex}")
else:
    print(f"... received response '{response}'")
```

Apple

```
[session.messaging sendRequest:[PTDiffusionPrimitive
 requestWithLongLong:42]
                        toPath:message_path
  int64NumberCompletionHandler:^(NSNumber *response, NSError *error)
{
    if (error) {
        NSLog(@"Failed to send to %@. Error: %@", message_path,
 error);
    } else {
        NSLog(@"Received response: %@", response);
    }
}];
```

Responding to request messages sent to a message path

Required permissions: send_to_session permission for the specified message path, register_handler permission, and view_session permission to register to receive session property values with the request message

Define a request handler to receive and respond to request messages that have a specific data type.

JavaScript

```
var jsonType = diffusion.datatypes.json();
var requestJson = jsonType.from({ "foo": "bar"});
var responseJson = jsonType.from({ "ying": "yang"});
// Define a request handler for json topic type
var handler = {
    onRequest: function(request, context, responder) {
        responder.respond(responseJson, jsonType);
    },
    onError: function() {},
    onClose: function() {}
};
```

.NET

```
/// <summary>
/// A simple IRequestHandler implementation that prints confirmation
 of the actions completed.
/// </summary>
internal class SimpleRequestHandler : IRequestHandler<string, string>
 {
    /// <summary>
    /// Indicates that the request handler was closed.
    /// </summary>
    public void OnClose()
        => WriteLine( "A request handler was closed." );

    /// <summary>
    /// Indicates that the request handler has received error.
    /// </summary>
    public void OnError( ErrorReason errorReason )
        => WriteLine( $"A request handler has received error:
 '{errorReason}'." );

    /// <summary>
    /// Indicates that a request was received and responds to it.
    /// </summary>
    /// <remarks>On invalid request you would call: <see
 cref="IResponder{TResponse}.Reject(string)"/>.</remarks>
    public void OnRequest( string request, IRequestContext context,
 IResponder<string> responder ) {
        WriteLine( $"Received request: '{request}'." );
        responder.Respond( DateTime.UtcNow.ToLongTimeString() );
    }
}
```

Java and Android

```
private final class JSONRequestHandler implements
 MessagingControl.RequestHandler<JSON, JSON> {
    @Override
    public void onClose() {
        ....
    }

    @Override
    public void onError(ErrorReason errorReason) {
        ....
    }

    @Override
    public void onRequest(JSON request, RequestContext context,
 Responder<JSON> responder) {
        ....
        responder.respond(response);
    }
}
```

Python

```
def callback(request: str, **kwargs) -> str:
```

```
    return f"Hello there, {request}!"
```

Apple

```
@interface NumberRequestDelegate :
 NSObject<PTDiffusionNumberRequestDelegate>
@end

@implementation NumberRequestDelegate

-(void)diffusionTopicTreeRegistration:
(PTDiffusionTopicTreeRegistration *)registration
          didReceiveRequestWithNumber:(nullable NSNumber *)number
                              context:(PTDiffusionRequestContext
 *)context
                            responder:(PTDiffusionResponder
 *)responder;
{
    // Do something when a request is received.
}

- (void)diffusionTopicTreeRegistration:(nonnull
 PTDiffusionTopicTreeRegistration *)registration
                       didFailWithError:(nonnull NSError *)error
{
    // Do something if the registration fails.
}

- (void)diffusionTopicTreeRegistrationDidClose:(nonnull
 PTDiffusionTopicTreeRegistration *)registration
{
    // Do something if the registration closes.
}
```

Register the request handler against a message path. You can only register one request handler against each message path.

JavaScript

```
var handler = {
    onRequest: function(request, context, responder) {},
    onError: function() {},
    onClose: function() {}
};
session.messages.addRequestHandler('topic', handler);
```

.NET

```
/// <summary>
/// Client implementation that registers a handler to listen for
 messages on a path.
/// </summary>
public sealed class ReceivingPathRequestMessages {

    public ReceivingPathRequestMessages(string serverUrl) {
        var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
```

```
            .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
            .Open(serverUrl);

        var messaging = session.Messaging;
        string messagingPath = ">random/requestResponse";

        var requestHandler = new SimpleRequestHandler();
        var requestHandlerRegistration = await
 messaging.AddRequestHandlerAsync(
            messagingPath, requestHandler );

        try {
            Thread.Sleep( 60000 );//wait for messages...
        } finally {
            // Close session
            await requestHandlerRegistration.CloseAsync();
            session.Close();
        }
    }
}
```

Java and Android

```
messagingControl.addRequestHandler(messagePath, JSON.class,
 JSON.class, new JSONRequestHandler());
```

Python

```
# Register handler to receive the request
handler = RequestHandler(
    callback,
    request_type=request_type,
    response_type=request_type
)
print("Registering request handler...")
try:
    await session.messaging.add_request_handler(path,
 handler=handler)
except diffusion.DiffusionError as ex:
    print(f"ERROR: {ex}")
else:
    print("... request handler registered")
```

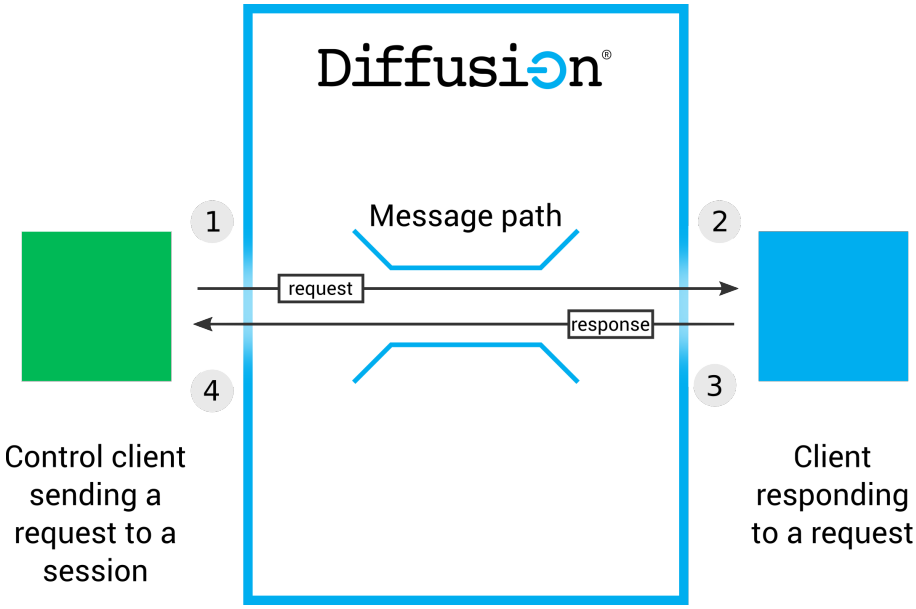
Apple

```
// Ensure to maintain a strong reference to your delegate as it
// is referenced weakly by the Diffusion client library.
NumberRequestDelegate *const delegate = [NumberRequestDelegate new];
PTDiffusionRequestHandler *const handler = [PTDiffusionPrimitive
 int64RequestHandlerWithDelegate:delegate];

[session.messagingControl addRequestHandler:handler
                                    forPath:path

 completionHandler:^(PTDiffusionTopicTreeRegistration *registration,
 NSError *error)
{
    // Check error is `nil`, indicating success.
```

```
}];
```

Sending request messages to a session

A client session can send a request message containing typed data directly to a client session. The receiving client session can then send a response message containing typed data. The request and response messages are addressed through the same message path.



When a request message is sent to a specific client session and that session responds, the following events occur:

1. A control client session sends a request message to a client session, specifying the message path to send the message through and the session ID of the client session to send the request message to.
2. The client session receives the request message through a request stream.
3. The client session uses a responder to send a response to the request message.
4. The control client session receives the response.

Both the request message and the response message contain typed values. The messages can contain data of one of the following types: JSON, binary, string, 64-bit integer, or double. The response message is not required to be the same data type as the request it responds to.

Sending a request to a session

Required permissions: send_to_session permission for the specified message path and register_handler permission

Usually, it is a control client session in your organization's backend that sends messages directly to other sessions.

Send the request message specifying the following information:

- The session ID of the client session to send the request to
- The message path to send the request and receive the response through
- The request message
- The datatype of the request message
- The datatype of the response message

JavaScript

```
control.messages.sendRequest('foo', 'Hello client', session_id,
 diffusion.datatypes.json(), diffusion.datatypes.json())
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Session;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that sends a request message to a
 filter,
    /// then sends another request directly to the session, and
 displays the response.
    /// </summary>
    public sealed class SendingSessionRequestMessages {
        private readonly string messagingPath = ">random/
requestResponse";

        public SendingSessionRequestMessages(string serverUrl) {
            string serverUrl = args[ 0 ];
            var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            var messaging = session.Messaging;
            var requestCallback = new RequestCallback();

            // Filter messaging is used to get the session ID for
 this example
            int requestsSent = await
 messaging.SendRequestToFilterAsync(
                "$Principal EQ 'client'",
```

```
                messagingPath,
                "Hello?",
                requestCallback);

            Thread.Sleep( 1000 );

            // Send message to a session using obtained session ID
            string response = await
 messaging.SendRequestAsync<string, string>(
                requestCallback.SessionId, messagingPath, "Time");
            WriteLine($"Received response: '{response}'.");

            // Close the session
            session.Close();
        }

        /// <summary>
        /// A simple IFilteredRequestCallback implementation that
 prints confirmation of the actions completed.
        /// </summary>
        private class RequestCallback :
 IFilteredRequestCallback<string> {
            public ISessionId SessionId { get; private set; }

            /// <summary>
            /// Indicates that the stream was closed.
            /// </summary>
            public void OnClose()
                => WriteLine( "A request handler was closed." );

            /// <summary>
            /// Indicates error received by the callback.
            /// </summary>
            public void OnError( ErrorReason errorReason )
                => WriteLine( $"A request handler has received error:
 '{errorReason}'." );

            /// <summary>
            /// Indicates that a response message was received.
            /// </summary>
            public void OnResponse(ISessionId sessionId, string
 response) => SessionId = sessionId;

            /// <summary>
            /// Indicates that a error response message was received.
            /// </summary>
            public void OnResponseError( ISessionId sessionId,
 Exception exception )
                => WriteLine( $"Response error received from session
 {sessionId}: '{exception}'." );
        }
    }
}
```

Java and Android

```
//Establish client session and control session
final Session control =
 Diffusion.sessions().principal("control").password("password").open("ws://
localhost:8080");
```

```
final Session client =
 Diffusion.sessions().principal("client").password("password").open("ws://
localhost:8080");

//Obtain the Messaging and MessagingControl features
final MessagingControl messagingControl =
 control.feature("MessagingControl.class");
final Messaging messaging = client.feature(Messaging.class);

//Create a JSON object to send as a request
final JSON request =
 Diffusion.dataTypes().json().fromJsonString("\"hello\"");

//Create a local request stream for the client to receive direct
 requests from the control session
messaging.setRequestStream("foo", JSON.class, JSON.class,
 requestStream);

//Send the request to a message path "foo" and wait for (at most) 5
 seconds until the response is received.
final JSON response =
 messagingControl.sendRequest(client.getSessionId(), "foo", request,
 JSON.class, JSON.class).get(5, TimeUnit.SECONDS);
```

Python

```
# Sending the request and receiving the response.
print(f"Sending request: '{request}' to session {session_id}...")
try:
    response = await session.messaging.send_request_to_session(
        path=path, session_id=session_id,
 request=request_type(request)
    )
except diffusion.DiffusionError as ex:
    print(f"ERROR: {ex}")
else:
    print(f"... received response '{response}'")
```

Apple

```
[session.messagingControl sendRequest:[PTDiffusionPrimitive
 requestWithLongLong:42]
                            toSessionId:sessionId
                                   path:message_path
         int64NumberCompletionHandler:^(NSNumber *response, NSError*
 error)
{
    if (error) {
        NSLog(@"Failed to send to %@. Error: %@", message_path,
 error);
    } else {
        NSLog(@"Received response: %@", response);
    }
}];
```

Responding to messages sent to a session

Required permissions: send_to_message_handler for the specified message path

Define a request stream to receive and respond to request messages that have a specific data type.

JavaScript

```
var handler = {
    onRequest : function(request, context, responder) {
        ....
        responder.respond(response);
    },
    onError : function(error) {},
    onClose : function() {}
}
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// A simple IRequestStream implementation that prints
 confirmation of the actions completed.
    /// </summary>
    internal class SimpleRequestStream : IRequestStream<string,
 string> {
        /// <summary>
        /// Indicates that the request stream was closed.
        /// </summary>
        public void OnClose()
            => WriteLine( "A request handler was closed." );

        /// <summary>
        /// Indicates that the request stream has received error.
        /// </summary>
        public void OnError( ErrorReason errorReason )
            => WriteLine( $"A request handler has received error:
 {errorReason}." );

        /// <summary>
```

```
        /// Indicates that a request was received and responds to it.
        /// </summary>
        /// <remarks>On invalid request you would call: <see
 cref="IResponder{TResponse}.Reject(string)"/>.</remarks>
        public void OnRequest( string path, string request,
 IResponder<string> responder ) {
            if ( request == "Hello?" ) {    // message to the filter
 to obtain the session ID
                responder.Respond( "Yes" );
            } else {
                WriteLine( $"Received request: '{request}'." );

 responder.Respond( DateTime.UtcNow.ToLongTimeString() );
            }
        }
    }
}
```

Java and Android

```

private final class JSONRequestStream implements
 Messaging.RequestStream<JSON, JSON> {

    @Override
    public void onClose() {
        ....
    }

    @Override
    public void onError(ErrorReason errorReason) {
        ....
    }

    @Override
    public void onRequest(String path, JSON request, Responder<JSON>
 responder) {
        ....
    }
}
```

Python

```
def callback(request: str, **kwargs) -> str:
    return f"Hello there, {request}!"
```

Apple

```

@interface NumberRequestStreamDelegate :
 NSObject<PTDiffusionNumberRequestStreamDelegate>
@end

@implementation NumberRequestStreamDelegate
- (void)    diffusionStream:(nonnull PTDiffusionStream *)stream
didReceiveRequestWithNumber:(nullable NSNumber *)number
                  responder:(nonnull PTDiffusionResponder *)responder
{
    // Do something when a request is received.
}
```

```
- (void)diffusionStream:(nonnull PTDiffusionStream *)stream
       didFailWithError:(nonnull NSError *)error
{
    // Do something if the stream fails.
}

- (void)diffusionDidCloseStream:(nonnull PTDiffusionStream *)stream
{
    // Do something if the stream closes.
}
```

Add the request stream against a message path. You can only add one request stream for each message path.

JavaScript

```
control.messages.setRequestStream("foo", diffusion.datatypes.json(),
 diffusion.datatypes.json(), request_stream);
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that registers a handler to listen for
 messages on a path.
    /// </summary>
    public sealed class ReceivingSessionRequestMessages {

        public ReceivingSessionRequestMessages(string serverUrl) {

            var session =
 Diffusion.Sessions.Principal( "client" ).Password( "password" ).Open( serverUrl
            var messaging = session.Messaging;
            string messagingPath = ">random/requestResponse";
```

```
            var requestStream = new SimpleRequestStream();
            messaging.SetRequestStream( messagingPath,
 requestStream );

            try
            {
                Thread.Sleep( 60000 );//wait for messages...
            }
            finally
            {
                // Close session
                messaging.RemoveRequestStream( messagingPath );
                session.Close();
            }
        }
    }
}
```

Java and Android

```
messaging.setRequestStream("foo", JSON.class, JSON.class,
 requestStream);
```

Python

```
# Register handler to receive the request
handler = RequestHandler(
    callback,
    request_type=request_type,
    response_type=request_type
)
session.messaging.add_stream_handler(path, handler=handler,
 addressed=True)
```

Apple

```
// Ensure to maintain a strong reference to your request stream as it
// is referenced weakly by the Diffusion client library.
NumberRequestStreamDelegate *delegate = [NumberRequestStreamDelegate
 new];
PTDiffusionRequestStream *requestStream = [PTDiffusionPrimitive
 int64RequestStreamWithDelegate:delegate];
[session.messaging setRequestStream:requestStream
 forPath:message_path];
```

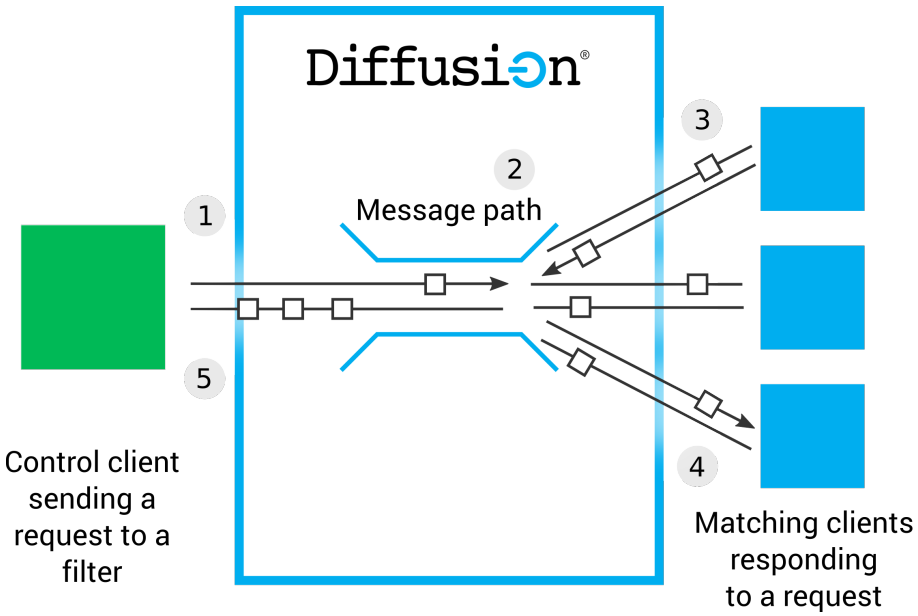
Sending request messages to a session filter

A client session can send a request message containing typed data directly to each client session in the set of connected client sessions that match a specified session properties filter. The receiving

client sessions can then send a response message containing typed data. The request and response messages are addressed through the same message path.

> **Note:** Sending messages to a set of client sessions defined by a filter is not intended for high throughput of data. If you have a lot of data to send or want to send data to a lot of client sessions, use the pub-sub capabilities of Diffusion. Subscribe the set of client sessions to a topic and publish the data as updates through that topic.

For more information about session properties and how to filter connected client sessions using their properties, see Session properties on page 220 and Session filtering on page 222.



When a request message is sent to a set of client sessions and those sessions respond, the following events occur:

1. A control client session sends a request message, specifying the filter that selects the client sessions to receive the request and specifying the message path to send the message through.
2. The Diffusion server evaluates the query and sends the message on to connected client sessions whose session properties match the filter
3. The client sessions in the filtered set each receive the request message through a request stream.
4. Each client session uses a responder to send a response to the request message.
5. The control client session receives responses from each of the clients sessions specified by the filter.

The request messages and the response messages contain typed values. The messages can contain data of one of the following types: JSON, binary, string, 64-bit integer, or double. The response messages are not required to be the same data type as the request or as the response messages from other client sessions.

Sending a request message to a filter

Required permissions: send_to_session permission for the specified message path and register_handler permission

Usually, it is a control client session in your organization's backend that sends messages to a filter. For more information about defining a session filter, see Session filtering on page 222.

Send the request message specifying the following information:

- The query to use to filter which client sessions to send the requests to

- The message path to send the request and receive the responses through
- The request message
- The datatype of the request message
- The datatype of the response message

JavaScript

```
var handler = {
    onResponse : function(sessionID, response) {},
    onResponseError : function(sessionID, error) {},
    onError : function(error) {}
}
control.messages.sendRequestToFilter(filter, 'foo', 'Hello clients',
 handler, diffusion.datatypes.json(), diffusion.datatypes.json());
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System;
using System.Threading;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features;
using PushTechnology.ClientInterface.Client.Session;
using static System.Console;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Client implementation that sends a request message to a
 filter and
    /// displays the response.
    /// </summary>
    public sealed class SendingFilterRequestMessages {
        private readonly string messagingPath = ">random/
requestResponse";

        public SendingFilterRequestMessages(string serverUrl) {

            var session =
 Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);
```

```
            var messaging = session.Messaging;
            var requestCallback = new RequestCallback();

            int requestsSent = await
 messaging.SendRequestToFilterAsync(
                "$Principal EQ 'client'",
                messagingPath,
                "Time",
                requestCallback );
            WriteLine( $"Sent request to {requestsSent} session(s)
 matching the filter." );

            // Close the session
            session.Close();
        }

        /// <summary>
        /// A simple IFilteredRequestCallback implementation that
 prints confirmation of the actions completed.
        /// </summary>
        private class RequestCallback :
 IFilteredRequestCallback<string> {
            /// <summary>
            /// Indicates that the stream was closed.
            /// </summary>
            public void OnClose()
                => WriteLine( "A request handler was closed." );

            /// <summary>
            /// Indicates error received by the callback.
            /// </summary>
            public void OnError( ErrorReason errorReason )
                => WriteLine( $"A request handler has received error:
 '{errorReason}'." );

            /// <summary>
            /// Indicates that a response message was received.
            /// </summary>
            public void OnResponse( ISessionId sessionId, string
 response )
                => WriteLine( $"Received response: '{response}'." );

            /// <summary>
            /// Indicates that a error response message was received.
            /// </summary>
            public void OnResponseError( ISessionId sessionId,
 Exception exception )
                => WriteLine( $"Response error received from session
 {sessionId}: '{exception}'." );
        }
    }
}
```

Java and Android

```
//Establish control sesssion
final Session control =
 Diffusion.sessions().principal("control").password("password").open("ws://
localhost:8080");
```

```
//Obtain the MessagingControl feature
final MessagingControl messagingControl =
 control.feature(MessagingControl.class);

//Create a JSON object to send as a request
final JSON request =
 Diffusion.dataTypes().json().fromJsonString("\"hello\"");

//Send the request to a message path "foo", to all sessions which
 do not have a 'control' principal and wait for (at most) 5 seconds
 until the response (number of responses) is received.
final int numberOfResponses =
 messagingControl.sendRequestToFilter("$Principal NE 'control'",
 "foo", request, JSON.class, JSON.class).get(5, TimeUnit.SECONDS);
```

Python

```
# sending the request and receiving the number of expected responses
print(f"Sending request: '{request}' to session filter
  `{session_filter}`...")
try:
    response = await session.messaging.send_request_to_filter(
        session_filter=session_filter,
        path=path,
        request=request_type(request),
    )
except diffusion.DiffusionError as ex:
    print(f"ERROR while sending request to session filter: {ex}")
else:
    print(f"... expecting {response} response(s) ...")
```

Responding to messages sent to a filter

Required permissions: send_to_message_handler for the specified message path

To the receiving client session, a request message sent to a filter is the same as a request message sent directly to the session. The receiving client session responds in the same way.

See Responding to messages sent to a session for details.

Authenticating new sessions

A client session can use the AuthenticationControl feature to authenticate other client sessions.

Registering a control authentication handler

Required permissions: authenticate, register_handler

A client can register an authentication handler that can be called when a client connects to the Diffusion server or changes the principal and credentials it is connected with.

The authentication handler can decide whether a client's authentication request is allowed or denied, or the authentication handler can abstain from the decision, in which case the next configured authentication handler is called.

A client can propose session properties when it connects. If the authentication handler allows a client's authentication request, it can choose to set the proposed properties.

The authentication handler can set any user-defined session property, and some fixed session properties, such as the `$Roles` session property.

For more information about authentication and role-based security, see Authentication on page 149.

Related concepts

Configuring authentication handlers on page 420
Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

Example: Register an authentication handler

The following examples use the Diffusion API to register a control authentication handler with the Diffusion server. The examples also include a simple or empty authentication handler.

The name by which the control authentication handler is registered must be configured in the `Server.xml` configuration file of the Diffusion server for the control authentication handler to be called to handle authentication requests.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */

using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Callbacks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Features.Control.Clients;
using PushTechnology.ClientInterface.Client.Security.Authentication;
using PushTechnology.DiffusionCore.Client.Types;
using static System.Console;
using PushTechnology.ClientInterface.Client.Session;
using System;

namespace PushTechnology.ClientInterface.Example {
    /// <summary>
    /// Implementation of a client which authenticates other sessions
 using
    /// a registered authentication handler.
    /// </summary>
    public sealed class AuthenticationControl{
        public AuthenticationControl(string serverUrl) {
```

```
            // Connect as a control session
            var session =
Diffusion.Sessions.Principal( "control" ).Password( "password" )
                .CertificateValidation((cert, chain, errors) =>
CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            WriteLine("Opening control session.");

            IRegistration registration = null;

            try
            {
                registration = await
session.AuthenticationControl.SetAuthenticationHandlerAsync(
                    "before-system-handler", new Authenticator(),
cancellationToken );

                WriteLine("Authentication handler registered.
Authenticator created.");

                Diffusion.Sessions.Principal("client")

.Credentials(Diffusion.Credentials.Password("password"))
                    .CertificateValidation((cert, chain, errors)
                        => CertificateValidationResult.ACCEPT)
                    .Open(serverUrl, new SessionOpenCallback());

                await Task.Delay(TimeSpan.FromMilliseconds(2000),
cancellationToken );
            } catch ( TaskCanceledException ) {
                //Task was cancelled;
            } finally {
                WriteLine("Closing control session.");

                await registration.CloseAsync();
                session.Close();
            }
        }

        /// <summary>
        /// Callback used when a session is opened using
ISessionFactory.Open
        /// </summary>
        private sealed class SessionOpenCallback :
ISessionOpenCallback
        {
            public void OnError(ErrorReason errorReason)
                => WriteLine($"An error occurred: {errorReason}");

            public void OnOpened(ISession session)
            {
                WriteLine("Other session opened.");

                session.Close();

                WriteLine("Other session closed.");
            }
        }

        /// <summary>
        /// Basic implementation of the control authenticator.
        /// </summary>
```

```
        private sealed class Authenticator : IControlAuthenticator {
            /// <summary>
            /// Method which decides whether a connection attempt
 should be allowed, denied or
            /// if another authenticator should evaluate this
 request.
            /// </summary>
            /// <param name="principal">The session principal.</
param>
            /// <param name="credentials">The credentials.</param>
            /// <param name="sessionProperties">The session
 properties.</param>
            /// <param name="proposedProperties">The client proposed
 properties.</param>
            /// <param name="callback">The callback.</param>
            public void Authenticate(
                string principal,
                ICredentials credentials,
                IReadOnlyDictionary<string, string>
 sessionProperties,
                IReadOnlyDictionary<string, string>
 proposedProperties,
                IAuthenticatorCallback callback ) {

                switch ( principal ) {
                case "admin": {
                        WriteLine( "Authenticator allowing connection
 with proposed properties." );
                        callback.Allow( proposedProperties );
                        break;
                    }
                case "client": {
                        WriteLine( "Authenticator allowing connection
 with no properties." );
                        callback.Allow();
                        break;
                    }
                case "block": {
                        WriteLine( "Authenticator denying
 connection." );
                        callback.Deny();
                        break;
                    }
                default: {
                        WriteLine( "Authenticator abstaining." );
                        callback.Abstain();
                        break;
                    }
                }
            }

            /// <summary>
            /// Notification of authenticator closure.
            /// </summary>
            public void OnClose() => WriteLine( "Authenticator
 closed." );

            /// <summary>
            /// Notification of error.
            /// </summary>
            /// <param name="errorReason">Error reason.</param>
            public void OnError( ErrorReason errorReason )
```

```
                => WriteLine( $"Authenticator received an error:
 {errorReason}" );
        }
    }
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import java.nio.charset.Charset;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.TimeUnit;

import com.pushtechnology.diffusion.client.Diffusion;
import com.pushtechnology.diffusion.client.callbacks.Stream;
import
 com.pushtechnology.diffusion.client.features.control.clients.AuthenticationCont
import
 com.pushtechnology.diffusion.client.features.control.clients.AuthenticationCont
import com.pushtechnology.diffusion.client.session.Session;
import com.pushtechnology.diffusion.client.types.Credentials;

/**
 * This is a control client which registers an authentication handler
 with a
 * Diffusion server.
 */
public final class ControlAuthenticationClient {

    /**
     * Main entry point for the control client.
     */
    public static void main(final String[] args) throws Exception {

        // The control client links to the server using the principal
 'admin', which is
        // authenticated by the system authentication handler (see
 etc/SystemAuthentication.store).
        // The principal must have REGISTER_HANDLER and AUTHENTICATE
 permissions.
        final Session session =
            Diffusion.sessions()
                .principal("admin")
                .password("password")
                .open("ws://diffusion.example.com:80");


 session.feature(AuthenticationControl.class).setAuthenticationHandler(
            "after-system-handler",
            new ExampleControlAuthenticationHandler()).get(10,
 TimeUnit.SECONDS);

        while (true) {
            Thread.sleep(60000);
        }
    }
```

```
   /**
    * An example of a control authentication handler.
    * <P>
    * This shows a simple example using a table of permitted
principals with
    * their passwords. It also demonstrates how the handler can
change the
    * properties of the client being authenticated.
    */
   private static class ExampleControlAuthenticationHandler
       extends Stream.Default
       implements ControlAuthenticator {

       private static final Map<String, byte[]> PASSWORDS = new
HashMap<>();
       static {
           PASSWORDS.put("manager",
"password".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("guest",
"asecret".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("brian",
"boru".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("another",
"apassword".getBytes(Charset.forName("UTF-8")));
       }

       @Override
       public void authenticate(
           String principal,
           Credentials credentials,
           Map<String, String> sessionProperties,
           Map<String, String> proposedProperties,
           Callback callback) {

           final byte[] passwordBytes = PASSWORDS.get(principal);

           if (passwordBytes != null &&
               credentials.getType() ==
Credentials.Type.PLAIN_PASSWORD &&
               Arrays.equals(credentials.toBytes(), passwordBytes))
{
               if ("manager".equals(principal)) {
                   // manager allows all proposed properties
                   callback.allow(proposedProperties);
               }
               else if ("brian".equals(principal)) {
                   // brian is allowed all proposed properties and
also gets
                   // the 'super' role added
                   final Map<String, String> result =
                       new HashMap<>(proposedProperties);
                   final Set<String> roles =
                       Diffusion.stringToRoles(
                           sessionProperties.get(Session.ROLES));
                   roles.add("super");
                   result.put(Session.ROLES,
Diffusion.rolesToString(roles));
                   callback.allow(result);
               }
               else {
                   // all others authenticated but ignoring proposed
properties
                   callback.allow();
```

```
                }
            }
            else {
                // Any principal not in the table is denied.
                callback.deny();
            }
        }
    }
}
```

C

```
/*
 * Diffusion can be configured to delegate authentication requests to
 * an external handler. This program provides an authentication
 * handler to demonstrate this feature. A detailed description of
 * security and authentication handlers can be found in the Diffusion
 * user manual.
 *
 * Authentication handlers are registered with a name, which is
 typically specified in
 * Server.xml
 *
 * Two handler names are provided by default;
 * before-system-handler and after-system-handler, and additional
 * handlers may be specified for Diffusion through the Server.xml
 file
 * and an accompanying Java class that implements the
 * AuthenticationHandler interface.
 *
 * This control authentication handler connects to Diffusion and
 attempts
 * to register itself with a user-supplied name, which should match
 the name
 * configured in Server.xml.
 *
 * The default behavior is to install as the "before-system-handler",
 * which means that it will intercept authentication requests before
 * Diffusion has a chance to act on them.
 *
 * It will:
 * <ul>
 * <li>Deny all anonymous connections</li>
 * <li>Allow connections where the principal and credentials (i.e.,
 username and password) match some hardcoded values</li>
 * <li>Abstain from all other decisions, thereby letting Diffusion
 and other authentication handlers decide what to do.</li>
 * </ul>
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include "diffusion.h"
#include "args.h"
#include "conversation.h"

struct user_credentials_s {
        const char *username;
```

```
        const char *password;
};

/*
 * Username/password pairs that this handler accepts.
 */
static const struct user_credentials_s USERS[] = {
        { "fish", "chips" },
        { "ham", "eggs" },
        { NULL, NULL }
};

ARG_OPTS_T arg_opts[] = {
        ARG_OPTS_HELP,
        {'u', "url", "Diffusion server URL", ARG_OPTIONAL,
 ARG_HAS_VALUE, "ws://localhost:8080"},
        {'n', "name", "Name under which to register the
 authentication handler", ARG_OPTIONAL, ARG_HAS_VALUE, "before-
system-handler"},
        {'p', "principal", "Principal (username) for the connection",
 ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        {'c', "credentials", "Credentials (password) for the
 connection", ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        END_OF_ARG_OPTS
};

/*
 * When the authentication service has been registered, this function
 will be
 * called.
 */
static int
on_registration(SESSION_T *session, void *context)
{
        printf("Registered authentication handler\n");
        return HANDLER_SUCCESS;
}

/*
 * When the authentication service has be deregistered, this function
 will be
 * called.
 */
static int
on_deregistration(SESSION_T *session, void *context)
{
        printf("Deregistered authentication handler\n");
        return HANDLER_SUCCESS;
}

/*
 * This is the function that is called when authentication has been
 delegated
 * from Diffusion.
 *
 * The response may return one of three values via the response
 parameter:
 * ALLOW:   The user is authenticated.
 * ALLOW_WITH_RESULT: The user is authenticated, and additional roles
 are
 *                    to be applied to the user.
 * DENY:    The user is NOT authenticated.
 * ABSTAIN: Allow another handler to make the decision.
```

```
 *
 * The handler should return HANDLER_SUCCESS in all cases, unless an
 actual
 * error occurs during the authentication process (in which case,
 * HANDLER_FAILURE is appropriate).
 */
static int
on_authentication(SESSION_T *session,
                  const SVC_AUTHENTICATION_REQUEST_T *request,
                  SVC_AUTHENTICATION_RESPONSE_T *response,
                  void *context)
{
        // No credentials, or not password type. We're not an
 authority for
        // this type of authentication so abstain in case some other
 registered
        // authentication handler can deal with the request.
        if(request->credentials == NULL) {
                printf("No credentials specified, abstaining\n");
                response->value = AUTHENTICATION_ABSTAIN;
                return HANDLER_SUCCESS;
        }
        if(request->credentials->type != PLAIN_PASSWORD) {
                printf("Credentials are not PLAIN_PASSWORD,
 abstaining\n");
                response->value = AUTHENTICATION_ABSTAIN;
                return HANDLER_SUCCESS;
        }

        printf("principal = %s\n", request->principal);
        printf("credentials = %*s\n",
               (int)request->credentials->data->len,
               request->credentials->data->data);

        if(request->principal == NULL || strlen(request->principal)
 == 0) {
                printf("Denying anonymous connection (no
 principal)\n");
                response->value = AUTHENTICATION_DENY; // Deny anon
 connections
                return HANDLER_SUCCESS;
        }

        char *password = malloc(request->credentials->data->len + 1);
        memmove(password, request->credentials->data->data, request-
>credentials->data->len);
        password[request->credentials->data->len] = '\0';

        int auth_decided = 0;
        int i = 0;
        while(USERS[i].username != NULL) {

                printf("Checking username %s vs %s\n", request-
>principal, USERS[i].username);
                printf("     and password %s vs %s\n", password,
 USERS[i].password);

                if(strcmp(USERS[i].username, request->principal) == 0
 &&
                   strcmp(USERS[i].password, password) == 0) {

                        puts("Allowed");
                        response->value = AUTHENTICATION_ALLOW;
```

```
                        auth_decided = 1;
                        break;
                }
                i++;

        }

        free(password);

        if(auth_decided == 0) {
                puts("Abstained");
                response->value = AUTHENTICATION_ABSTAIN;
        }

        return HANDLER_SUCCESS;
}

int
main(int argc, char** argv)
{
        HASH_T *options = parse_cmdline(argc, argv, arg_opts);
        if (options == NULL || hash_get(options, "help") != NULL) {
                show_usage(argc, argv, arg_opts);
                return EXIT_FAILURE;
        }

        char *url = hash_get(options, "url");
        char *name = hash_get(options, "name");
        char *principal = hash_get(options, "principal");
        char *credentials = hash_get(options, "credentials");

        /*
         * Create a session with Diffusion.
         */
        puts("Creating session");
        DIFFUSION_ERROR_T error = { 0 };
        SESSION_T *session = session_create(url,
                                            principal,
                                            credentials != NULL ?
 credentials_create_password(credentials) : NULL,
                                            NULL, NULL,
                                            &error);
        if (session == NULL) {
                fprintf(stderr, "TEST: Failed to create session\n");
                fprintf(stderr, "ERR : %s\n", error.message);
                return EXIT_FAILURE;
        }

        /*
         * Provide a set (via a hash map containing keys and NULL
         * values) to indicate what information about the connecting
         * client that we'd like Diffusion to send us.
         */
        HASH_T *detail_set = hash_new(5);
        char buf[2];
        sprintf(buf, "%d", SESSION_DETAIL_SUMMARY);
        hash_add(detail_set, strdup(buf), NULL);
        sprintf(buf, "%d", SESSION_DETAIL_LOCATION);
        hash_add(detail_set, strdup(buf), NULL);
        sprintf(buf, "%d", SESSION_DETAIL_CONNECTOR_NAME);
        hash_add(detail_set, strdup(buf), NULL);

        /*
```

```
         * Register the authentication handler.
         */
        AUTHENTICATION_REGISTRATION_PARAMS_T auth_registration_params
= {
                .name = name,
                .detail_set = detail_set,
                .on_registration = on_registration,
                .authentication_handlers.on_authentication =
on_authentication
        };

        puts("Sending registration request");
        SVC_AUTHENTICATION_REGISTER_REQUEST_T *reg_request =
                authentication_register(session,
auth_registration_params);

        /*
         *  Wait a while before moving on to deregistration.
         */
        sleep(30);

        AUTHENTICATION_DEREGISTRATION_PARAMS_T
auth_deregistration_params = {
                .on_deregistration = on_deregistration,
                .original_request = reg_request
        };

        /*
         * Deregister the authentication handler.
         */
        printf("Deregistering authentication handler\n");
        authentication_deregister(session,
auth_deregistration_params);

        session_close(session, NULL);
        session_free(session);

        return EXIT_SUCCESS;
}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Related concepts

Configuring authentication handlers on page 420
Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

Developing a control authentication handler

Implement the `ControlAuthenticator` interface to create a control authentication handler.

About this task

This example demonstrates how to implement a control authentication handler in Java.

Procedure

1. Edit the `etc/Server.xml` configuration file to include a name that the control authentication handler can register with.

   Include the `control-authentication-handler` element in the list of authentication handlers. The order of the list defines the order in which the authentication handlers are called. The value of the `handler-name` attribute is the name that your control authentication handler registers as. For example:

   ```
   <security>
     <authentication-handlers>
       <-- Include a local authentication handler that can
    authenticate the control client -->
       <authentication-handler class="com.example.LocalHandler" />

       <-- Register your control authentication handler -->
       <control-authentication-handler handler-name="before-system-
   handler" />

     </authentication-handlers>
   </security>
   ```

   The client that registers your control authentication handler must first authenticate with the Diffusion server. Configure a local authentication handler that allows the client to connect.

2. Start the Diffusion server.

   - On UNIX®-based systems, run the `diffusion.sh` command in the *`diffusion_installation_dir`*`/bin` directory.
   - On Windows systems, run the `diffusion.bat` command in the *`diffusion_installation_dir`*`\bin` directory.

3. Create a simple client that registers a control authentication handler with the Diffusion server. You will need to create a Java class that implements `ControlAuthenticator`.

   ```
   package com.pushtechnology.diffusion.examples;

   import java.nio.charset.Charset;
   import java.util.Arrays;
   import java.util.HashMap;
   import java.util.Map;
   import java.util.Set;
   import java.util.concurrent.TimeUnit;

   import com.pushtechnology.diffusion.client.Diffusion;
   import com.pushtechnology.diffusion.client.callbacks.Stream;
   import
    com.pushtechnology.diffusion.client.features.control.clients.AuthenticationCo
   import
    com.pushtechnology.diffusion.client.features.control.clients.AuthenticationCo
   import com.pushtechnology.diffusion.client.session.Session;
   import com.pushtechnology.diffusion.client.types.Credentials;

   public final class ControlAuthenticationClient {

       private ControlAuthenticationClient() {
       }

       /**
        * Main entry point for the control client.
        */
       // CHECKSTYLE.OFF: UncommentedMain
   ```

```
   public static void main(final String[] args) throws Exception {

       // The control client connects to the server using the
principal 'admin'
       // which is authenticated by the system authentication
handler (see
       // etc/SystemAuthentication.store).
       // The principal must have REGISTER_HANDLER and
AUTHENTICATE permissions.
       final Session session =
           Diffusion.sessions()
               .principal("admin")
               .password("password")
               .open("ws://diffusion.example.com:80");


session.feature(AuthenticationControl.class).setAuthenticationHandler(
           "after-system-handler",
           new ExampleControlAuthenticationHandler()).get(10,
TimeUnit.SECONDS);

       while (true) {
           Thread.sleep(60000);
       }
   }


   /**
    * An example of a control authentication handler.
    * <p>
    * This shows a simple example using a table of permitted
principals with
    * their passwords. It also demonstrates how the handler can
change the
    * properties of the client being authenticated.
    */
   private static class ExampleControlAuthenticationHandler
       extends Stream.Default
       implements ControlAuthenticator {

       private static final Map<String, byte[]> PASSWORDS = new
HashMap<>();
       static {
           PASSWORDS.put("manager",
"password".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("guest",
"asecret".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("brian",
"boru".getBytes(Charset.forName("UTF-8")));
           PASSWORDS.put("another",
"apassword".getBytes(Charset.forName("UTF-8")));
       }

       @Override
       public void authenticate(
           String principal,
           Credentials credentials,
           Map<String, String> sessionProperties,
           Map<String, String> proposedProperties,
           Callback callback) {

           final byte[] passwordBytes = PASSWORDS.get(principal);
```

```
            if (passwordBytes != null &&
                credentials.getType() ==
Credentials.Type.PLAIN_PASSWORD &&
                Arrays.equals(credentials.toBytes(),
passwordBytes)) {
                if ("manager".equals(principal)) {
                    // manager allows all proposed properties
                    callback.allow(proposedProperties);
                }
                else if ("brian".equals(principal)) {
                    // brian is allowed all proposed properties and
also gets
                    // the 'super' role added
                    final Map<String, String> result =
                        new HashMap<>(proposedProperties);
                    final Set<String> roles =
                        Diffusion.stringToRoles(
                            sessionProperties.get(Session.ROLES));
                    roles.add("super");
                    result.put(Session.ROLES,
Diffusion.rolesToString(roles));
                    callback.allow(result);
                }
                else {
                    // all others authenticated but ignoring
proposed properties
                    callback.allow();
                }
            }
            else {
                // Any principal not in the table is denied.
                callback.deny();
            }
        }
    }
}
```

4. Start your client.

   It connects to the Diffusion server and registers the control authentication handler with the name `after-system-handler`.

Results

When a client authenticates, the Diffusion server forwards the authentication request to the authentication handler you have registered. Your authentication handler can ALLOW, DENY, or ABSTAIN from the authentication decision. If your authentication handler returns an ALLOW or DENY decision, this decision is used as the response to the authenticating client. If your authentication handler returns an ABSTAIN decision, the Diffusion server forwards the authentication request to the next authentication handler. For more information, see Authentication on page 149.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

Authentication on page 149
You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

Related tasks

Developing a local authentication handler on page 391

Implement the `Authenticator` interface to create a local authentication handler.

Developing a composite authentication handler
Developing a composite control authentication handler

Updating the system authentication store

A client can use the SystemAuthenticationControl feature to update the system authentication store. The information in the system authentication store is used by the system authentication handler to authenticate users and assign roles to them.

Querying the store

Required permissions: view_security

The client can get a snapshot of the current information in the system authentication store. This information is returned as an object model.

Updating the store

Required permissions: modify_security

The client can use a command script to update the system authentication store. The command script is a string that contains a command on each line. These commands are applied to the current state of the system authentication store.

The update is transactional. Unless all of the commands in the script can be applied, none of them are.

Using a script builder

You can use a script builder to create the command script used to update the system authentication store. Use the script builder to create commands for the following actions:

- Set the authentication decision for anonymous principals
- Add principals to the store
- Delete principals from the store
- Change the password of a principal
- Assign roles to principals
- Accept client-proposed session properties

Related reference

System authentication handler on page 154
Diffusion provides an authentication handler that uses principal, credential, and roles information stored in the Diffusion server to make its authentication decision.

DSL syntax: system authentication store

The scripts that you can use with the SystemAuthenticationControl feature to update the system authentication store are formatted according to a domain-specific language (DSL). You can use the

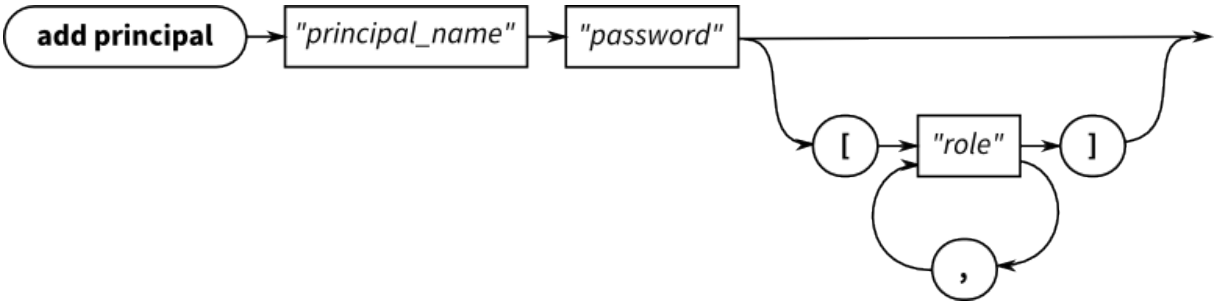
script builders provided in the APIs to create a script to update the system authentication store. However, if you want to create the script by some other method, ensure that it conforms to the DSL.

> **Note:** You should stop the server before editing the system authentication store directly. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

The following sections each describe the syntax for a single line of the file.

Adding a principal

Railroad diagram



Backus-Naur form

`add principal` " *principal_name* " " *password* " `[ '[' '"` *role* `"[ , "` *role* `"] ']' ]`

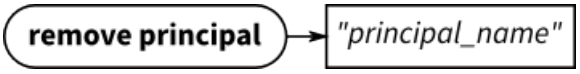
Example

```
add principal "user6" "passw0rd"
add principal "user13" "passw0rd" ["CLIENT", "TOPIC_CONTROL"]
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Removing a principal

Railroad diagram



Backus-Naur form

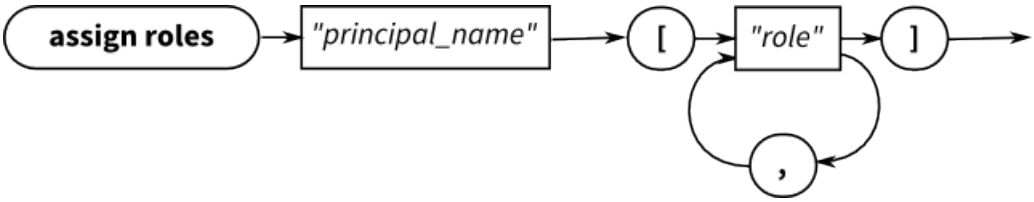
`remove principal` " *principal_name* "

Example

```
remove principal "user25"
```

Assigning roles to a principal

Railroad diagram



Backus-Naur form

`assign roles` " *principal_name* " `'[' '"` *role* `"[ , "` *role* `"] ']'`

Example

```
assign roles "agent77" ["CLIENT", "CLIENT_CONTROL"]
```

When you use this command to assign roles to a principal, it overwrites any existing roles assigned to that principal. Ensure that all the roles you want the principal to have are listed in the command.

Setting the password for a principal

Railroad diagram

Backus-Naur form

`set password` " *principal_name* " " *password* "

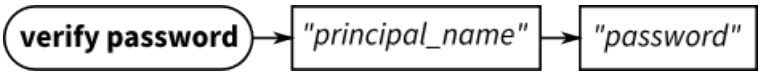
Example

```
set password "user1" "passw0rd"
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Verifying the password for a principal

Railroad diagram



Backus-Naur form

`verify password` " *principal_name* " " *password* "

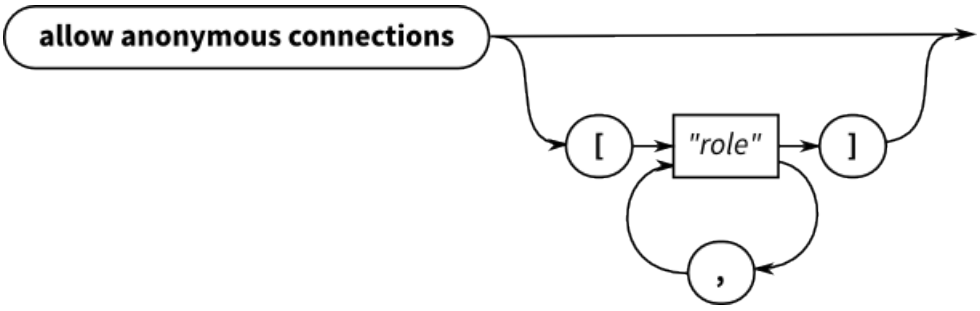
Example

```
verify password "user1" "passw0rd"
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Allowing anonymous connections

Railroad diagram



Backus-Naur form

`allow anonymous connections['['` " *role* `"[ ,` " *role* `"]']']`

Example

```
allow anonymous connections [ "CLIENT" ]
```

Denying anonymous connections

Railroad diagram

Backus-Naur form

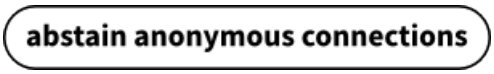
```
deny anonymous connections
```

Example

```
deny anonymous connections
```

Abstaining from providing a decision about anonymous connections

Railroad diagram



Backus-Naur form

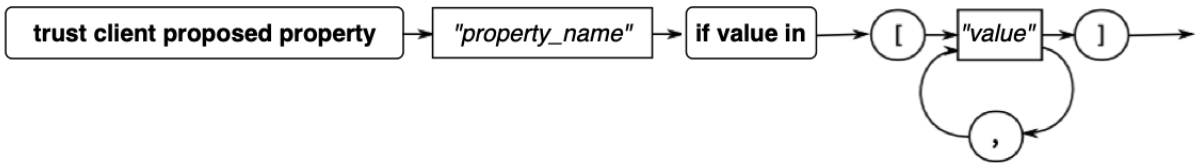
```
abstain anonymous connections
```

Example

```
abstain anonymous connections
```

Accepting client-proposed session properties with approved values

Railroad diagram



Backus-Naur form

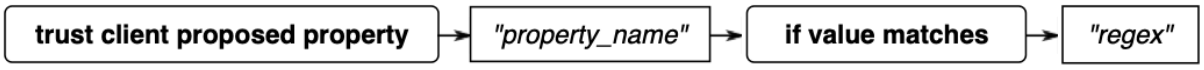
```
trust client proposed property "property_name" '['"value"[ ,"value"]']'
```

Example

```
trust client proposed property "Foo" if value in ["x", "y", "z"]
```

Accepting client-proposed session properties matching a regex

Railroad diagram



Backus-Naur form

```
trust client proposed property "property_name" if value matches "regex"
```

Example

```
trust client proposed property "Foo" if value matches "^\d{3}-?
\d{2}-?\d{4}$"
```

Use Java-style regular expressions. Evaluation uses `java.util.regex.Pattern`.

Removing a previously-declared trusted client-proposed session property

Railroad diagram

Backus-Naur form

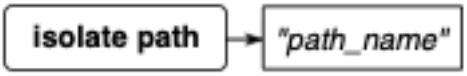
`ignore client proposed property "` *property_name* `"`

Example

```
ignore client proposed property "Foo"
```

Isolating a path from permissions inheritance

Railroad diagram



Backus-Naur form

`isolate path "` *path_name* `"`

Example

```
isolate path "foo/bar/baz"
```

Example: Update the system authentication store

The following examples use the SystemAuthenticationControl feature in the Diffusion API to update the system authentication store.

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Types;
using static System.Console;

namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Client implementation that demonstrates how to update the
 system authentication store.
    /// </summary>
    public sealed class SystemAuthenticationControl
    {
        public SystemAuthenticationControl(string serverUrl)
        {
            // Connect as an admin session
            var session =
 Diffusion.Sessions.Principal("admin").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);

            string testPrincipal = "TestPrincipal";

            // Create a new principal

            try
            {
                WriteLine($"Creating principal '{testPrincipal}'.");

                string storeScript =
 session.SystemAuthenticationControl.Script
                    .AddPrincipal(testPrincipal, "password", new
 List<string>())
                    .TrustClientProposedPropertyIn("Foo", new
 List<string> { "value1", "value2" })
                    .TrustClientProposedPropertyMatches("Bar",
 "regex1")
                    .ToScript();

                await
 session.SystemAuthenticationControl.UpdateStoreAsync(storeScript);

                WriteLine($"'{testPrincipal}' has been created.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to create principal : {ex}.");
            }

            //Assign roles to the principal

            try
            {
                WriteLine($"Adding the roles of Administrator and
 Modify Session to '{testPrincipal}'.");

                string script1 =
 session.SystemAuthenticationControl.Script
```

```
                    .AssignRoles(testPrincipal, new[]
 { "ADMINISTRATOR", "MODIFY_SESSION" })
                    .ToScript();

                await
 session.SystemAuthenticationControl.UpdateStoreAsync(script1);

                WriteLine($"Roles have been added.");
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to assign roles : {ex}.");
            }
            finally
            {
                session.Close();
            }
        }
    }
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import java.util.HashSet;
import java.util.Set;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.client.features.control.clients.SystemAuthenticati
import
 com.pushtechnology.diffusion.client.features.control.clients.SystemAuthenticati
import com.pushtechnology.diffusion.client.session.Session;

/**
 * An example of using a control client to alter the system
 authentication
 * configuration.
 * <P>
 * This uses the {@link SystemAuthenticationControl} feature only.
 *
 * @author Push Technology Limited
 * @since 5.2
 */
public class ControlClientChangingSystemAuthentication {

    private static final Logger LOG =
        LoggerFactory.getLogger(
            ControlClientChangingSystemAuthentication.class);

    private final SystemAuthenticationControl
 systemAuthenticationControl;
    private final ScriptBuilder emptyScript;

    /**
```

```
     * Constructor.
     */
    public ControlClientChangingSystemAuthentication() {

        final Session session = Diffusion.sessions()
            // Authenticate with a user that has the VIEW_SECURITY
and
            // MODIFY_SECURITY permissions.
            .principal("admin").password("password")
            // Use a secure channel because we're transferring
sensitive
            // information.
            .open("wss://diffusion.example.com:80");

        systemAuthenticationControl =
            session.feature(SystemAuthenticationControl.class);
        emptyScript = systemAuthenticationControl.scriptBuilder();
    }

    /**
     * For all system users, update the assigned roles to replace the
     * "SUPERUSER" role and with "ADMINISTRATOR".
     *
     * @return a CompletableFuture that completes when the operation
succeeds or
     *         fails.
     *
     *         <p>
     *         If the operation was successful, the CompletableFuture
will
     *         complete successfully.
     *
     *         <p>
     *         Otherwise, the CompletableFuture will complete
exceptionally with
     *         an {@link ExecutionException}. See
     *         {@link
SystemAuthenticationControl#getSystemAuthentication()} and
     *         {@link
SystemAuthenticationControl#updateStore(String)} for
     *         common failure reasons.
     */
    public CompletableFuture<Void> changeSuperUsersToAdministrators()
{

        return systemAuthenticationControl
            .getSystemAuthentication()
            .thenCompose(configuration -> {

                final String script = configuration

                    // For each principal ...
                    .getPrincipals().stream()

                    // ... that has the SUPERUSER assigned role ...
                    .filter(p ->
p.getAssignedRoles().contains("SUPERUSER"))

                    // ... create a script that updates the assigned
roles to
                    // replace SUPERUSER with ADMINISTRATOR ...
                    .map(p -> {
                        final Set<String> newRoles =
```

```
                            new HashSet<>(p.getAssignedRoles());
                        newRoles.remove("SUPERUSER");
                        newRoles.add("ADMINISTRATOR");
                        return emptyScript.assignRoles(p.getName(),
 newRoles);
                    })

                    // ... create a single combined script.
                    .reduce(emptyScript, (sb1, sb2) ->
 sb1.append(sb2))
                    .script();

                LOG.info("Sending the following script to the server:
\n{}",
                    script);

                return
 systemAuthenticationControl.updateStore(script)
                    // Convert CompletableFuture<?> to
                    // CompletableFuture<Void>.
                    .thenAccept(ignored -> { });
            });
    }

    /**
     * Close the session.
     */
    public void close() {
        systemAuthenticationControl.getSession().close();
    }
}
```

C

```

/*
 * This examples demonstrates how to interact with the system
 * authentication store.
 */

#include <stdio.h>

#include <apr.h>
#include <apr_thread_mutex.h>
#include <apr_thread_cond.h>

#include "diffusion.h"
#include "args.h"
#include "service/svc-system-auth-control.h"

apr_pool_t *pool = NULL;
apr_thread_mutex_t *mutex = NULL;
apr_thread_cond_t *cond = NULL;

ARG_OPTS_T arg_opts[] = {
        ARG_OPTS_HELP,
        {'u', "url", "Diffusion server URL", ARG_OPTIONAL,
 ARG_HAS_VALUE, "ws://localhost:8080"},
        {'p', "principal", "Principal (username) for the connection",
 ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
```

```
        {'c', "credentials", "Credentials (password) for the
 connection", ARG_OPTIONAL, ARG_HAS_VALUE, NULL},
        END_OF_ARG_OPTS
};

/*
 * This callback is invoked when the system authentication store is
 * received, and prints the contents of the store.
 */
int
on_get_system_authentication_store(SESSION_T *session,
                                   const
 SYSTEM_AUTHENTICATION_STORE_T store,
                                   void *context)
{
        puts("on_get_system_authentication_store()");

        printf("Got %ld principals\n", store.system_principals-
>size);

        char **names = get_principal_names(store);
        for(char **name = names; *name != NULL; name++) {
                printf("Principal: %s\n", *name);

                char **roles = get_roles_for_principal(store, *name);
                for(char **role = roles; *role != NULL; role++) {
                    printf("  |- Role: %s\n", *role);
                }
                free(roles);
        }
        free(names);

        switch(store.anonymous_connection_action) {
        case ANONYMOUS_CONNECTION_ACTION_ALLOW:
                puts("Allow anonymous connections");
                break;
        case ANONYMOUS_CONNECTION_ACTION_DENY:
                puts("Deny anonymous connections");
                break;
        case ANONYMOUS_CONNECTION_ACTION_ABSTAIN:
                puts("Abstain from making anonymous connection
 decision");
                break;
        }

        puts("Anonymous connection roles:");
        char **roles = get_anonymous_roles(store);
        for(char **role = roles; *role != NULL; role++) {
                printf("  |- Role: %s\n", *role);
        }
        free(roles);

        apr_thread_mutex_lock(mutex);
        apr_thread_cond_broadcast(cond);
        apr_thread_mutex_unlock(mutex);

        return HANDLER_SUCCESS;
}

int
main(int argc, char **argv)
{
        /*
```

```
         * Standard command-line parsing.
         */
        const HASH_T *options = parse_cmdline(argc, argv, arg_opts);
        if(options == NULL || hash_get(options, "help") != NULL) {
                show_usage(argc, argv, arg_opts);
                return EXIT_FAILURE;
        }

        const char *url = hash_get(options, "url");
        const char *principal = hash_get(options, "principal");
        CREDENTIALS_T *credentials = NULL;
        const char *password = hash_get(options, "credentials");
        if(password != NULL) {
                credentials = credentials_create_password(password);
        }

        /*
         * Setup for condition variable
         */
        apr_initialize();
        apr_pool_create(&pool, NULL);
        apr_thread_mutex_create(&mutex, APR_THREAD_MUTEX_UNNESTED,
pool);
        apr_thread_cond_create(&cond, pool);

        /*
         * Create a session with Diffusion.
         */
        SESSION_T *session;
        DIFFUSION_ERROR_T error = { 0 };
        session = session_create(url, principal, credentials, NULL,
NULL, &error);
        if(session == NULL) {
                fprintf(stderr, "TEST: Failed to create session\n");
                fprintf(stderr, "ERR : %s\n", error.message);
                return EXIT_FAILURE;
        }

        /*
         * Request the system authentication store.
         */
        const GET_SYSTEM_AUTHENTICATION_STORE_PARAMS_T params = {
                .on_get = on_get_system_authentication_store
        };

        apr_thread_mutex_lock(mutex);

        get_system_authentication_store(session, params);

        apr_thread_cond_wait(cond, mutex);
        apr_thread_mutex_unlock(mutex);

        /*
         * Close the session and tidy up.
         */
        session_close(session, NULL);
        session_free(session);

        apr_thread_mutex_destroy(mutex);
        apr_thread_cond_destroy(cond);
        apr_pool_destroy(pool);
        apr_terminate();
```

```
        return EXIT_SUCCESS;
}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Updating the security store

A client can use the SecurityControl feature to update the security store. The information in the security store is used by the Diffusion server to define the permissions assigned to roles and the roles assigned to anonymous sessions and named sessions.

Querying the store

Required permissions: view_security

The client can get a snapshot of the current information in the security store. This information is returned as an object model.

Updating the store

Required permissions: modify_security

The client can use a command script to update the security store. The command script is a string that contains a command on each line. These commands are applied to the current state of the security store.

The update is transactional. Unless all of the commands in the script can be applied, none of them are.

As of Diffusion 6.5, updates to read_topic permission assignments are applied immediately to all sessions. There is no need to reauthenticate a session or reassign a role.

Using a script builder

You can use a script builder to create the command script used to update the security store. Use the script builder to create commands for the following actions:

- Set the global permissions assigned to a named role
- Set the default path permissions assigned to a named role
- Set the path permissions associated with a specific path assigned to a named role

  This can include explicitly setting a role to have no permissions at a path.
- Remove the path permissions associated with a specific path assigned to a named role
- Set the roles included in a named role
- Set the roles assigned to sessions authenticated with a named principal
- Set the roles assigned to anonymous sessions

Related reference

Configuring security on page 446

The `Security.store` file contains the rules that determine the permissions for each role. The `SystemAuthentication.store` stores the principals recognised by the system authentication handler.

DSL syntax: security store

The scripts that you can use with the SecurityControl feature to update the security store are formatted according to a domain-specific language (DSL). You can use the script builders provided in the APIs to create a script to update the security store. However, if you want to create the script by some other method, ensure that it conforms to the DSL.

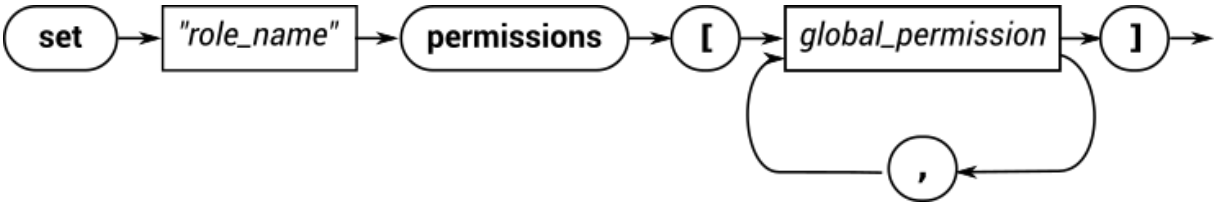
> **Note:** You should stop the server before editing the security store directly. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

The following sections each describe the syntax for a single line of the script file.

> **Note:** The **path** keyword is synonymous with the **topic** keyword used in previous releases of Diffusion. Both keywords are accepted. Prefer **path**.

Assigning global permissions to a role

Railroad diagram



Backus-Naur form

`set "` *role_name* `" permissions ['[' global_permission [ , global_permission ] ']']`

Example

```
set "ADMINISTRATOR" permissions [CONTROL_SERVER, VIEW_SERVER,
 VIEW_SECURITY, MODIFY_SECURITY]
set "CLIENT_CONTROL" permissions [VIEW_SESSION, MODIFY_SESSION,
 REGISTER_HANDLER]
```

Assigning default path permissions to a role

Railroad diagram

Backus-Naur form

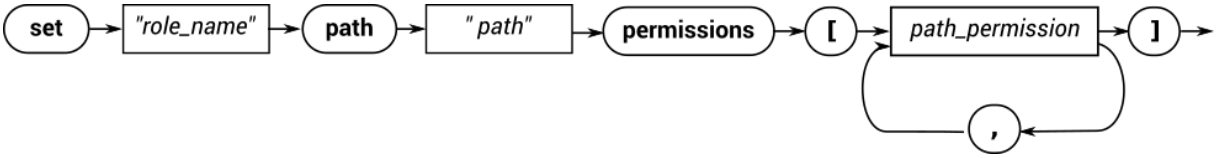
`set "` *role_name* `" default path permissions ['[' path_permission [ , path_permission ] ']']`

Example

```
set "CLIENT" default path permissions [READ_TOPIC ,
 SEND_TO_MESSAGE_HANDLER]
```

Assigning path permissions associated with a specific path to a role

Railroad diagram



Backus-Naur form

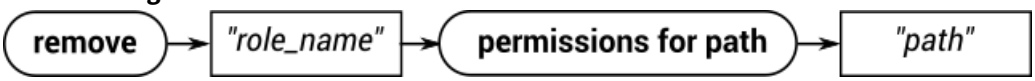
`set "` *role_name* `" path "` *path* `" permissions ['[' ` *path_permission* `[ , ` *path_permission* `]']']`

Example

```
set "CLIENT" path "foo/bar" permissions [READ_TOPIC,
 SEND_TO_MESSAGE_HANDLER]
set "ADMINISTRATOR" path "foo" permissions [ MODIFY_TOPIC ]
set "CLIENT_CONTROL" path "foo" permissions [ ]
```

Removing all path permissions associated with a specific path to a role

Railroad diagram



Backus-Naur form

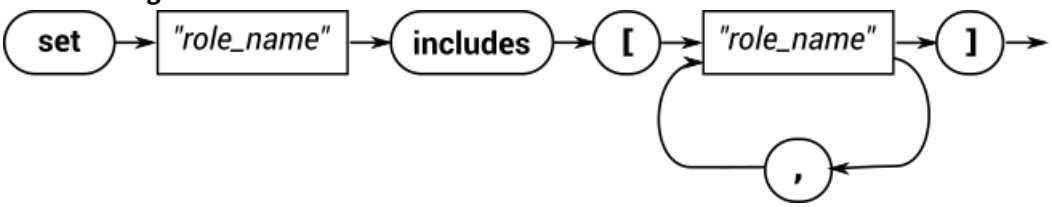
`remove "` *role_name* `" permissions for path "` *path* `"`

Example

```
remove "CLIENT" permissions for path "foo/bar"
```

Including roles within another role

Railroad diagram



Backus-Naur form

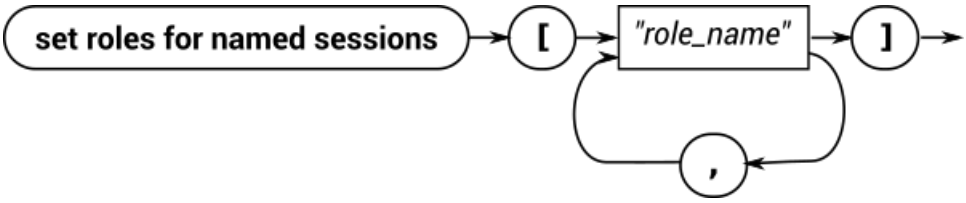
`set "` *role_name* `" includes ['[' "` *role_name* `"[ , "` *role_name* `"]']']`

Example

```
set "ADMINISTRATOR" includes ["CLIENT_CONTROL" , "TOPIC_CONTROL"]
set "CLIENT_CONTROL" includes ["CLIENT"]
```

Assigning roles to a named session

Railroad diagram



Backus-Naur form

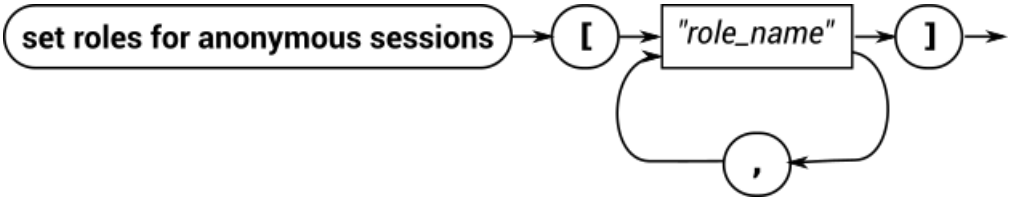
```
set roles for named sessions['['"role_name"[ ,"role_name"]']']
```

Example

```
set roles for named sessions ["CLIENT"]
```

Assigning roles to an anonymous session

Railroad diagram



Backus-Naur form

```
set roles for anonymous sessions['['"role_name"[ ,"role_name"]']']
```

Example

```
set roles for anonymous sessions ["CLIENT"]
```

Example: Update the security store

The following examples use the SecurityControl feature in the Diffusion API to update the security store.

JavaScript

```
// Session security allows you to change the principal that a session
 is authenticated as. It also  allows users to
// query and update server-side security and authentication stores,
 which control users, roles and permissions.
// This enables you to manage the capabilities that any logged in
 user will have access to.

// Connect to Diffusion with control client credentials
diffusion.connect({
    host   : 'diffusion.example.com',
    port   : 443,
    secure : true,
    principal : 'control',
    credentials : 'password'
}).then(function(session) {

    // 1. A session change their principal by re-authenticating
    session.security.changePrincipal('admin',
 'password').then(function() {
```

```
        console.log('Authenticated as admin');
    });

    // 2. The security configuration provides details about roles and
 their assigned permissions
    session.security.getSecurityConfiguration().then(function(config)
 {
        console.log('Roles for anonymous sessions: ',
 config.anonymous);
        console.log('Roles for named sessions: ', config.named);
        console.log('Available roles: ', config.roles);
    }, function(error) {
        console.log('Unable to fetch security configuration', error);
    });

    // 3. Changes to the security configuration are done with a
 SecurityScriptBuilder
    var securityScriptBuilder =
 session.security.securityScriptBuilder();

    // Set the permissions for a particular role - global and topic-
scoped
    // Each method on a script builder returns a new builder
    var setPermissionScript =
 securityScriptBuilder.setGlobalPermissions('SUPERUSER',
 ['REGISTER_HANDLER'])

 .setTopicPermissions('SUPERUSER', '/foo', ['UPDATE_TOPIC'])
                                                    .build();

    // Update the server-side store with the generated script

 session.security.updateSecurityStore(setPermissionScript).then(function()
 {
        console.log('Security configuration updated successfully');
    }, function(error) {
        console.log('Failed to update security configuration: ',
 error);
    });

    // 4. The system authentication configuration lists all users &
 roles

 session.security.getSystemAuthenticationConfiguration().then(function(config)
 {
        console.log('System principals: ', config.principals);
        console.log('Anonymous sessions: ', config.anonymous);
    }, function(error) {
        console.log('Unable to fetch system authentication
 configuration', error);
    });

    // 5. Changes to the system authentication config are done with a
 SystemAuthenticationScriptBuilder
    var authenticationScriptBuilder =
 session.security.authenticationScriptBuilder();

    // Add a new user and set password & roles.
    var addUserScript =
 authenticationScriptBuilder.addPrincipal('Superman',
 'correcthorsebatterystapler')

 .assignRoles('Superman', ['SUPERUSER'])
```

```
                                                        .build();

    // Update the system authentication store

 session.security.updateAuthenticationStore(addUserScript).then(function()
 {
        console.log('Updated system authentication config');
    }, function(error) {
        console.log('Failed to update system authentication: ',
 error);
    });
});
```

.NET

```
/**
 * Copyright © 2021 Push Technology Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
 * See the License for the specific language governing permissions
 and
 * limitations under the License.
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using PushTechnology.ClientInterface.Client.Factories;
using
 PushTechnology.ClientInterface.Client.Features.Control.Clients.SecurityControl;
using PushTechnology.ClientInterface.Client.Session;
using PushTechnology.ClientInterface.Client.Types;
using static System.Console;

namespace PushTechnology.ClientInterface.Example
{
    /// <summary>
    /// Client implementation that demonstrates how to update the
 security store.
    /// </summary>
    public sealed class SecurityControl
    {
        public SecurityControl(string serverUrl)
        {
            // Connect as an admin session
            var session =
 Diffusion.Sessions.Principal("admin").Password("password")
                .CertificateValidation((cert, chain, errors) =>
 CertificateValidationResult.ACCEPT)
                .Open(serverUrl);
```

```
            string role = "ADMINISTRATOR";

            IReadOnlyCollection<GlobalPermission> defaultPermissions
= null;
            ISecurityConfiguration securityConfig = null;

            try
            {
                //Get the default global permissions for the Admin
role.
                securityConfig = await
session.SecurityControl.GetSecurityAsync();

                var adminRole = securityConfig.Roles.Where(x =>
x.Name == role).FirstOrDefault();
                defaultPermissions = adminRole.GlobalPermissions;

                WriteLine($"The Administrator role has the following
global permissions by default:");

                foreach (var permission in defaultPermissions)
                {
                    WriteLine($"'{permission}'");
                }
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to get global permissions :
{ex}.");
            }

            try
            {
                //Add the following global permissions for the Admin
role.
                var permissions = new
List<GlobalPermission>(defaultPermissions);
                permissions.AddRange(new[]
{ GlobalPermission.REGISTER_HANDLER,

GlobalPermission.VIEW_SESSION });

                WriteLine($"Adding further permissions...");

                string script =

session.SecurityControl.Script.SetGlobalPermissions(role,
permissions).ToScript();

                await
session.SecurityControl.UpdateStoreAsync(script);
            }
            catch (Exception ex)
            {
                WriteLine($"Failed to set global permissions :
{ex}.");
            }
            finally
            {
                session.Close();
            }
        }
    }
```

```
}
```

Java and Android

```
package com.pushtechnology.diffusion.examples;

import static java.util.Collections.emptySet;
import static java.util.stream.Collectors.toCollection;

import java.util.Set;
import java.util.TreeSet;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.pushtechnology.diffusion.client.Diffusion;
import
 com.pushtechnology.diffusion.client.features.control.clients.SecurityControl;
import
 com.pushtechnology.diffusion.client.features.control.clients.SecurityControl.Ro
import
 com.pushtechnology.diffusion.client.features.control.clients.SecurityControl.Sc
import
 com.pushtechnology.diffusion.client.features.control.clients.SecurityControl.Se
import com.pushtechnology.diffusion.client.session.Session;
import com.pushtechnology.diffusion.client.types.PathPermission;

/**
 * An example of using a control client to alter the security
 configuration.
 * <P>
 * This uses the {@link SecurityControl} feature only.
 *
 * @author Push Technology Limited
 * @since 5.3
 */
public class ControlClientChangingSecurity {

    private static final Logger LOG =
        LoggerFactory.getLogger(
            ControlClientChangingSecurity.class);

    private final SecurityControl securityControl;
    private final ScriptBuilder emptyScript;

    /**
     * Constructor.
     */
    public ControlClientChangingSecurity() {

        final Session session = Diffusion.sessions()
            // Authenticate with a user that has the VIEW_SECURITY
 and
            // MODIFY_SECURITY permissions.
            .principal("admin").password("password")
            // Use a secure channel because we're transferring
 sensitive
            // information.
            .open("wss://diffusion.example.com:80");
```

```
        securityControl = session.feature(SecurityControl.class);
        emptyScript = securityControl.scriptBuilder();
    }

    /**
     * This will update the security store to ensure that all roles
start with a
     * capital letter (note that this does not address changing the
use of the
     * roles in the system authentication store).
     *
     * @return a CompletableFuture that completes when the operation
succeeds or
     *         fails.
     *
     *         <p>
     *         If the operation was successful, the CompletableFuture
will
     *         complete successfully.
     *
     *         <p>
     *         Otherwise, the CompletableFuture will complete
exceptionally with
     *         an {@link ExecutionException}. See
     *         {@link SecurityControl#getSecurity()} and
     *         {@link SecurityControl#updateStore(String)} for common
failure
     *         reasons.
     */
    public CompletableFuture<Void> capitalizeRoles() {
        return
securityControl.getSecurity().thenCompose(this::capitalizeRoles);
    }

    private CompletableFuture<Void> capitalizeRoles(
        SecurityConfiguration configuration) {

        final String script = emptyScript

            .setRolesForAnonymousSessions(

capitalizeSet(configuration.getRolesForAnonymousSessions()))

            .setRolesForNamedSessions(

capitalizeSet(configuration.getRolesForNamedSessions()))

            .append(configuration
                // For each role ...
                .getRoles().stream()
                // ... build a script that capitalises that role ...
                .map(this::capitalizeRole)
                /// .. and combine the per-role scripts into one.
                .reduce(emptyScript, (sb1, sb2) -> sb1.append(sb2)))

            .script();

        LOG.info("Sending the following script to the server:\n{}",
script);

        return securityControl.updateStore(script)
            // Convert CompletableFuture<?> to
CompletableFuture<Void>.
```

```
            .thenAccept(ignored -> { });
    }

    private ScriptBuilder capitalizeRole(Role role) {
        final String oldName = role.getName();
        final String newName = capitalizeString(oldName);

        ScriptBuilder builder = emptyScript;

        // Only if new name is different
        if (!oldName.equals(newName)) {
            if (!role.getGlobalPermissions().isEmpty()) {
                builder = builder
                    // Remove global permissions for old role
                    .setGlobalPermissions(oldName, emptySet())
                    // Set global permissions for new role
                    .setGlobalPermissions(
                        newName, role.getGlobalPermissions());
            }

            if (!role.getDefaultPathPermissions().isEmpty()) {
                builder = builder
                    // Remove default path permissions for old role
                    .setDefaultPathPermissions(oldName, emptySet())
                    // Set default path permissions for new role
                    .setDefaultPathPermissions(
                        newName, role.getDefaultPathPermissions());
            }

            builder = builder.append(
                role.getPathPermissions().entrySet().stream().map(
                    entry -> {
                        final String path = entry.getKey();
                        final Set<PathPermission> permissions =
entry.getValue();

                        return emptyScript
                            // Remove path permissions for old role
                            .removePathPermissions(oldName, path)
                            // Set path permissions for new role
                            .setPathPermissions(newName, path,
permissions);
                    })
                    .reduce(emptyScript, (sb1, sb2) ->
sb1.append(sb2)));
        }

        final Set<String> oldIncludedRoles = role.getIncludedRoles();

        if (oldIncludedRoles.isEmpty()) {
            return builder;
        }

        return builder
            // Remove old included roles.
            .setRoleIncludes(oldName, emptySet())

            // Set new roles even if role name did not change as the
included
            // roles may be changed.
            .setRoleIncludes(newName,
capitalizeSet(oldIncludedRoles));
    }
```

```
    private static Set<String> capitalizeSet(Set<String> roles) {
        return roles.stream()
            .map(ControlClientChangingSecurity::capitalizeString)
            .collect(toCollection(TreeSet::new));
    }

    private static String capitalizeString(String role) {
        return Character.toUpperCase(role.charAt(0)) +
 role.substring(1);
    }

    /**
     * Close the session.
     */
    public void close() {
        securityControl.getSession().close();
    }
}
```

Change the URL from that provided in the example to the URL of the Diffusion server.

Managing sessions

A client session with the appropriate permissions can receive notifications and information about other client sessions. A client session with the appropriate permissions can also manage other client sessions.

Closing client sessions

Required permissions: view_session, modify_session

A client can close any client session, providing the requesting client knows the session ID of the target client.

Requests are routed to the correct server within a Diffusion server cluster.

A client can close all sessions specified by a session filter.

If applied to a server cluster, this closes sessions matching the filter on all the servers.

.NET

```
var _clientControl = session.ClientControl;
_clientControl.Close( sessionID, callback );
```

Java and Android

```
ClientControl clientControl = session.feature(ClientControl.class);
clientControl.close(sessionID,callback);
```

Changing security roles

Required permissions: view_session, modify_session

A client can change the security roles of another session identified by session ID, or a group of sessions that matches a session filter expression.

Requests identified by session ID are routed to the correct server within a Diffusion server cluster; requests using a session filter are applied to sessions matching the filter on all the servers in a cluster.

Java and Android

```
CompletableFuture<?> changeRoles(SessionId sessionId,
    Set<String> rolesToRemove,
    Set<String> rolesToAdd)
    throws IllegalArgumentException
```

Related concepts

Session properties on page 220
A client session has a number of properties associated with it. Properties are key-value pairs. Both the key and the value are case sensitive.

Session filtering on page 222
Session filters enable you to query the set of connected client sessions on the Diffusion server based on their session properties.

Working with session properties

A client session with the appropriate permissions can view, request, or update the session properties of another client session.

Session properties

Each client session has a number of properties associated with it. Properties are keys and values. Both the key and the value are case sensitive. These session properties can be used by other clients to select sets of client session to perform actions on.

For more information, see Session properties on page 220.

Receiving notifications of client session events and their session properties

Required permissions: view_session, register_handler

To receive notifications when any client session opens, closes, or is updated, register a listener to listen for these events:

JavaScript

```
const listener = {
    onSessionOpen: (event) => {
        // The action to take on a client session open notification
    },
    onSessionClose: (event) => {
        // The action to take on a client session close notification
    },
    onSessionEvent: (event) => {
        // The action to take on a client session update notification
    },
    onActive: () => {
```

```
        // The listener is active
    },
    onClose: () => {
        // The listener is closed
    },
    onError: (err) => {
        // An error occurred
    },
};

// Register a listener for session properties
try {
    await
 session.clients.setSessionPropertiesListener(diffusion.clients.PropertyKeys.ALL
 listener);
} catch (err) {
    console.log('An error occurred when trying to set the listener');
}
```

.NET

```
var _clientControl = session.ClientControlGet;

// Set up a listener to receive notification of all sessions
_clientControl.SetSessionPropertiesListener( propertyListener,
 "$Country", "Department" );
```

Java and Android

```
ClientControl clientControl = session.feature(ClientControl.class);

clientControl.setSessionPropertiesListener(
    new ClientControl.SessionPropertiesListener.Default() {
        @Override
        public void onSessionOpen(){
            // The action to take on a client session open
 notification
        }
        @Override
        public void onSessionEvent(){
            // The action to take on a client session update
 notification
        }
        @Override
        public void onSessionClose(){
            // The action to take on a client session close
 notification
        }
    },
    // The session properties to receive
    "$Country", "$Department");
```

C

```
/*
* Register a session properties listener.
```

```
*
* Requests all "fixed" properties, i.e. those defined by
* Diffusion rather than user-defined properties.
*/
SET_T *required_properties = set_new_string(5);
set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_FIXED_PROPERTIES);

// Set the parameters to callbacks previously defined
SESSION_PROPERTIES_REGISTRATION_PARAMS_T params = {
      .on_registered = on_registered,
      .on_registration_error = on_registration_error,
      .on_session_open = on_session_open,
      .on_session_close = on_session_close,
      .on_session_update = on_session_update,
      .on_session_error = on_session_error,
      .required_properties = required_properties
};
session_properties_listener_register(session, params);
```

When registering this listener, specify which session properties to receive for each client session:

JavaScript

```
// Receive all fixed properties
session.clients.setSessionPropertiesListener(diffusion.clients.PropertyKeys.ALL_
 listener)
    .then(function() {

     });
// OR
// Receive all user-defined properties
session.clients.setSessionPropertiesListener(diffusion.clients.PropertyKeys.ALL_
 listener)
    .then(function() {

    });
// OR
// Receive all properties
session.clients.setSessionPropertiesListener([diffusion.clients.PropertyKeys.ALL
 diffusion.clients.PropertyKeys.ALL_USER_PROPERTIES], listener)
   .then(function() {

    });
```

.NET

```
// Define individual session properties to receive
_clientControl.SetSessionPropertiesListener(propertiesListener,
 "$Country", "Department" );
// OR
// Receive all fixed properties
_clientControl.SetSessionPropertiesListener(propertiesListener,
 SessionControlConstants.AllFixedProperties );
// OR
// Receive all user-defined properties
_clientControl.SetSessionPropertiesListener(propertiesListener,
 SessionControlConstants.AllUserProperties );
```

Java and Android

```
// Define individual session properties to receive
clientControl.setSessionPropertiesListener(
    new ClientControl.SessionPropertiesListener.Default() {
       // Define callbacks
    },
    "$Country", "$Department");
// OR
// Receive all fixed properties
clientControl.setSessionPropertiesListener(
    new ClientControl.SessionPropertiesListener.Default() {
       // Define callbacks
    },
    Session.ALL_FIXED_PROPERTIES);
// OR
// Receive all user-defined properties
clientControl.setSessionPropertiesListener(
    new ClientControl.SessionPropertiesListener.Default() {
       // Define callbacks
    },
    Session.ALL_USER_PROPERTIES);
```

C

```
// Receive all fixed properties
SET_T *required_properties = set_new_string(5);
set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_FIXED_PROPERTIES);

SESSION_PROPERTIES_REGISTRATION_PARAMS_T params = {
      //Other parameters
      .required_properties = required_properties
};
// OR
// Receive all user-defined properties
SET_T *required_properties = set_new_string(5);
set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_USER_PROPERTIES);

SESSION_PROPERTIES_REGISTRATION_PARAMS_T params = {
      //Other parameters
      .required_properties = required_properties
};
// OR
// Receive all properties
SET_T *required_properties = set_new_string(5);
set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_FIXED_PROPERTIES);
set_add(required_properties,
 PROPERTIES_SELECTOR_ALL_USER_PROPERTIES);

SESSION_PROPERTIES_REGISTRATION_PARAMS_T params = {
      //Other parameters
      .required_properties = required_properties
};
```

When the listening client first registers a listener, it receives a notification for every client session that is currently open. When subsequent client sessions open, the listening client receives a notification for those clients.

When the listening client is notified of a session event, it receives the requested session properties as a map of keys and values.

When the listening client is notified of a session closing, it also receives the reason that the session was closed. If the client session becomes disconnected from the Diffusion server, the listener might not receive notification of session close immediately. If reconnection is configured for the client, when the client disconnects, its session goes into reconnecting state for the configured time (the default is 60 seconds) before going into a closed state.

Getting properties of specific client sessions

Required permissions: view_session

A client can make an asynchronous request the session properties of any client session from the Diffusion server, providing the requesting client knows the session ID of the target client.

JavaScript

```
// Get fixed session properties
 session.clients.getSessionProperties(sessionID,
 diffusion.clients.PropertyKeys.ALL_FIXED_PROPERTIES)
     .then(function{

     });
```

.NET

```
var _clientControl = session.ClientControl;
_clientControl.GetSessionProperties( sessionID,
 SessionControlConstants.AllFixedProperties,
 sessionPropertiesCallback );
```

Java and Android

```
// Get fixed session properties
ClientControl clientControl = session.feature(ClientControl.class);
clientControl.getSessionProperties(sessionID,
 Session.ALL_FIXED_PROPERTIES, sessionPropertiesCallback);
```

C

```
GET_SESSION_PROPERTIES_PARAMS_T params = {
        .session_id = session_id,
        .required_properties = properties,
        .on_session_properties = on_session_properties
};

get_session_properties(session, params);
```

Update the properties of specific client sessions

Required permissions: view_session, modify_session

A client session with the appropriate permissions can update the value of existing user-defined session properties or add new user-defined properties for any client session or set of client sessions.

As part of the update session properties request, provide a map of the keys for the session properties you want to update or add and the new values. If you provide a `null` value for a session property, that property is deleted from the session. A successful update session properties request returns a map of the updated properties and their old values.

Specify a single session to change the user-defined session properties for the session by providing the session ID.

If you have a cluster of Diffusion servers, requests specifying a session ID can be routed to the server within the cluster which hosts that session.

Java and Android

```
// Change the session properties of a single session
ClientControl clientControl = session.feature(ClientControl.class);
final CompletableFuture<Map<String, String>> result =
 clientControl.setSessionProperties(sessionID, map_of_properties);
```

Specify a set of client sessions to change the user-defined session properties for by providing a filter query expression. For more information about filter query expressions, see Session filtering on page 222.

If you have a cluster of Diffusion servers, the change of session properties will be applied to sessions which match the filter on all servers within the cluster.

Java and Android

```
// Change the session properties of set of sessions defined by a
 filter expression
ClientControl clientControl = session.feature(ClientControl.class);
final CompletableFuture<Map<String, String>> result =
 clientControl.setSessionProperties(filter, map_of_properties);
```

Handling client queues

Each client session has a queue on the Diffusion server. Messages to be sent to the client are queued here. You can monitor the state of these queues and set client queue behavior.

Receiving notifications of client queue events

Required permissions: view_session, register_handler

A client can register a handler that is notified when outbound client queues at the Diffusion server reach pre-configured thresholds.

Java and Android

```
ClientControl clientControl = session.feature(ClientControl.class);
clientControl.setQueueEventHandler(
```

```
 new ClientControl. QueueEventHandler.Default {

        @Override
        public void onUpperThresholdCrossed(
            final SessionId client,
            final MessageQueuePolicy policy) {

            // The action to perform when the queue upper threshold
 is crossed.
        }

        @Override
        public void onLowerThresholdCrossed(
            final SessionId client,
            final MessageQueuePolicy policy) {

            // The action to perform when the queue lower threshold
 is crossed.
        }
    }
);
```

Flow control

A client application rapidly making thousands of calls to the Diffusion server might overflow the internal queues for the client, which results in that client session being closed. Flow control automatically protects against these queues overflowing by progressively delaying messages from the client to the Diffusion server.

Supported platforms: Android, Java, .NET, and C

The flow control mechanism is a feature of the Diffusion client libraries that works automatically to protect the following internal queues for an individual client from overflowing:

- The outbound queue on the client where messages are queued to be sent to the Diffusion server
- The queue on the Diffusion server where responses to service requests are queued.

If these queues overflow, the client session is terminated.

Flow control is intended to benefit those clients that send a lot of data to the Diffusion server – for example, updates to publish to topics. Usually, these clients are those located in the back-end of your solution that perform control functions.

When is flow control enabled?

The client determines whether to enable flow control and the amount of delay to introduce into the client processing based on a calculated value called back pressure. Back pressure is calculated using the following criteria:

- Depth of the outbound client queue
- The number of pending responses to service requests
- Whether the current active thread is a callback thread

Back pressure can have a value between 0.0 and 1.0. 1.0 is the maximum amount of back pressure. The method used to calculate back pressure might be subject to change in future releases.

Flow control introduces sleeps into the client processing. The length of these sleeps depends on the value of the back pressure. The maximum amount of delay introduced into client processing by flow control is 100 ms. The amount of delay introduced by flow control might be subject to change in future releases.

The flow control behavior of a client cannot be configured.

How to tell that flow control is enabled

When flow control is enabled for a Android, Java, or .NET client, the client logs messages at DEBUG level. The client logs each time a delay is introduced. The log message has the following form:

```
2016-09-26 11:15:48,344 DEBUG [PushConnectorPool-thread-18]
 c.p.d.f.SleepingFlowControl(apply) - pressure=1.0 => sleep for 100
 ms
```

The log message includes the current back pressure and the length of delay introduced.

The C client does not log its flow control behavior.

Actions to take

Diffusion clients can occasionally become flow controlled in response to very heavy load or unusual network conditions. However, if your clients are constantly being flow controlled, your Diffusion solution might not be correctly configured for the traffic load.

Consider taking the following actions:

- In your client design, ensure that if you have many requests to make to the Diffusion server that these requests are made from an application thread instead of a callback thread. Less flow control is applied when the active thread is a callback thread. For more information, see Best practice for developing clients on page 162.
- Ensure that your Diffusion server can handle the incoming messages from the clients. The default memory configuration might be causing the JVM running the Diffusion server to spend a lot of time in GC. For more information about tuning your JVM, see Memory considerations.
- Increase the maximum queue size on the connector your client uses. This can be configured for individual connectors in the `Connectors.xml` configuration file or as a default value for all connectors in the `Server.xml` configuration file. For more information, see Connectors.xml on page 438 and Server.xml on page 423.

Configuring conflation

Use the `CONFLATION` topic property to select a conflation policy for a topic.

Conflation settings

Conflation is configured through several settings.

Conflation can be enabled or disabled on a per-session basis using the `setConflated` API call. You can enable or disable conflation for multiple sessions using `setConflated` with a session filter.

> **Note:**
>
> `setConflated` is cluster-aware: it can route requests for a particular session ID to the correct server in a cluster. If you use a session filter, configuration changes will be applied to matching sessions on all servers within a cluster.

The default state for whether conflation is enabled is set using the `conflates` value in the `queue-definition` section of `Server.xml`. This is set to `true` for a fresh installation.

You can specify the conflation policy for each topic when it is created, using the `CONFLATION` topic property. The default policy is "conflate" (see below).

Conflation policies

Use the `CONFLATION` topic property to set a topic's conflation policy using one of the following values.

Table 34: Conflation topic properties

Property	Description
conflate (default)	Only conflate topics when a session with a full queue receives a new message.
always	An 'eager' policy that conflates updates as they arrive. If enabled, the queue will only ever contain one update.
off	Disable conflation for this topic. Sessions receive every update.
unsubscribe	When a session with a full queue receives a new message, unsubscribe the session with unsubscribe reason BACK_PRESSURE.

Related concepts

Conflation on page 102
Conflation of messages is the facility to reduce the amount of information sent to clients by combining or discarding updates.

Using conflation on page 102
You can configure how and when conflation is applied to different topics.

Logging from the client

Ensuring that your Diffusion client logs messages to inform of events and errors can be a valuable tool in developing and maintaining your clients.

Logging in JavaScript

The JavaScript client library logs messages to the console.

Log levels

Events are logged at different levels of severity. The log levels, ordered from most severe to least severe, are as follows:

Table 35: Log levels

Level	Description
`error`	Events that indicate a failure.
`warn`	Events that indicate a problem with operation.
`info`	Significant events.
`debug`	Verbose logging. Not usually enabled for production.

Level	Description
`trace`	High-volume logging of interest only to Push Technology Support. Push Technology Support may occasionally ask you to enable this log level to diagnose issues.

Configuring logging in the JavaScript client

You can use the JavaScript API to enable and configure logging at runtime.

```
diffusion.log(level)
```

To disable logging at runtime, set the level to `silent`.

```
diffusion.log('silent')
```

Note: Do not enable logging in your production clients. Use logging only during development of your clients.

Logging in Apple

The Apple client logs messages to the Apple system log facility.

Log levels

Events are logged at different levels of severity. The log levels, ordered from most severe to least severe, are as follows:

Table 36: Log levels

Level	Description
ERROR	Events that indicate a failure.
WARN	Events that indicate a problem with operation.
INFO	Significant events.
DEBUG	Verbose logging. Not usually enabled for production.
TRACE	High-volume logging of interest only to Push Technology Support. Push Technology Support may occasionally ask you to enable this log level to diagnose issues.

Configuring logging in the Apple client

You can use the Apple API to enable and configure logging at runtime.

```
PTDiffusionLogging *const l = [PTDiffusionLogging logging];

// Enable logging in the client library
l.enabled = YES;

// Change the level that the client logs at
l.level = [PTDiffusionLoggingLevel trace];
```

Note: Do not enable logging in your production clients. Use logging only during development of your clients.

Logging in Android

The Android client uses slf4j-android-logger to log messages to the Android logging system.

Log levels

Events are logged at different levels of severity. The log levels, ordered from most severe to least severe, are as follows:

Table 37: Log levels

Level	Description
ERROR	Events that indicate a failure.
WARN	Events that indicate a problem with operation.
INFO	Significant events.
DEBUG	Verbose logging. Not usually enabled for production.
TRACE	High-volume logging of interest only to Push Technology Support. Push Technology Support may occasionally ask you to enable this log level to diagnose issues.

Configuring logging in the Android client

The Android JAR, `diffusion-android-x.x.x.jar`, contains a properties file, `logger.properties`. Edit this properties file to configure logging in the Diffusion Android client.

The default `logger.properties` file contains the following properties:

```
de.psdev.slf4j.android.logger.logTag=DiffusionAndroidClient
de.psdev.slf4j.android.logger.defaultLogLevel=INFO
```

For more information about slf4j-android-logger, see https://github.com/PSDev/slf4j-android-logger

Logging in Java

The Java client uses SLF4J to log messages. Provide a bindings library to implement the SLF4J API and log out the messages.

Log levels

Events are logged at different levels of severity. The log levels, ordered from most severe to least severe, are as follows:

Table 38: Log levels

Level	Description
ERROR	Events that indicate a failure.
WARN	Events that indicate a problem with operation.
INFO	Significant events.
DEBUG	Verbose logging. Not usually enabled for production.

Level	Description
TRACE	High-volume logging of interest only to Push Technology Support. Push Technology Support may occasionally ask you to enable this log level to diagnose issues.

Configuring logging in the Java client

The Java JAR, `diffusion-client-x.x.x.jar`, uses the SLF4J API to log messages. It does not include an implementation that outputs the log messages.

Many SLF4J implementations are available.

1. Choose your preferred SLF4J implementation.
2. Ensure that the bindings JAR is on the classpath of your Java client.
3. Configure the SLF4J implementation to provide the logging behavior you require.

Log4j2

Log4j2 is a third-party SLF4J implementation provided by the Apache Software Foundation. For more information, see http://logging.apache.org/log4j/2.x/.

You can configure your Java clients to use log4j2 by completing the following steps:

1. Get the log4j2 bindings libraries.

   The JAR files can be downloaded from https://logging.apache.org/log4j/2.0/download.html.

   The log4j2 JAR files are also located in the `lib/thirdparty` directory of the Diffusion installation.
2. Ensure that the `log4j-api.jar` and `log4j-core.jar` files are on the client classpath.
3. Create a configuration file and ensure that is present on the client classpath.

   The following example `log4j2.xml` file outputs the log messages to a rolling set of files:

```
<Configuration status="warn" name="DiffusionClient">

    <Properties>
        <Property name="my.log.dir">../logs</Property>

        <!-- The log directory can be be overridden using the
 system property 'my.log.dir'. -->
        <Property name="log.dir">${sd:my.log.dir}</Property>

        <Property name="pattern">%date{yyyy-MM-dd HH:mm:ss.SSS}|
%level|%thread|%marker|%replace{%msg}{\|}{}|%logger%n%xEx
        </Property>
    </Properties>

    <Appenders>

        <RollingRandomAccessFile name="file" immediateFlush="false"
 fileName="${log.dir}/client.log"
            filePattern="${log.dir}/$${date:yyyy-MM}/client-%d{MM-
dd-yyyy}-%i.log.gz">

            <PatternLayout pattern="${pattern}" />

            <Policies>
                <OnStartupTriggeringPolicy />
                <TimeBasedTriggeringPolicy />
                <SizeBasedTriggeringPolicy size="250 MB" />
```

```
            </Policies>

            <DefaultRolloverStrategy max="20" />
        </RollingRandomAccessFile>
    </Appenders>

    <Loggers>
        <AsyncRoot level="info" includeLocation="false">
            <AppenderRef ref="file" />
        </AsyncRoot>
    </Loggers>
</Configuration>
```

For more information about configuring log4j2, see https://logging.apache.org/log4j/2.0/manual/configuration.html.

Logging in .NET

The .NET API produces logging information. The logging facility uses NLog.

NLog is included in the .NET client assembly. For more information about logging with NLog, see https://github.com/NLog/NLog/wiki/.

Logging basics

The NLog `Logger` class acts as the source of the log messages. In general, use one logger per class and pass in the name of the class as the logger name.

To log a message at a certain level use the write methods provided by the `Logger` class.

The following log levels are provided:

- Fatal
- Error
- Warn
- Info
- Debug
- Trace

These levels are listed in order from most severe to least severe.

Configuring the log output

You must configure NLog to output the log messages produced by your application to a target or targets. NLog provides a large number of targets for your output, including File and Console. NLog also enables you to create your own targets.

In addition to specifying targets for the log output, use NLog configuration to define rules that specify to which target log messages with particular levels or logger names are directed.

You can configure NLog in the following ways:

- Using a configuration file, `NLog.config`, that is located in the same directory as your client application.

  For more information, see https://github.com/NLog/NLog/wiki/Configuration-file.
- Using the Configuration API to configure NLog in your application code.

  For more information, see https://github.com/NLog/NLog/wiki/Configuration-API.

Logging in C

The C API provides no integrated logging feature. You can log out from your client code using your preferred method or framework.

Developing other components

Diffusion provides Java APIs that enable you to customize the behavior of your Diffusion server and related components.

Local authentication handlers

You can implement authentication handlers that authenticate client connections to the Diffusion server.

A local authentication handler is an implementation of the `Authenticatior` interface. Local authentication handlers can be implemented only in Java. The class file that contains a local authentication handler must be located on the classpath of the Diffusion server.

Related concepts

Configuring authentication handlers on page 420
Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

Developing a local authentication handler

Implement the `Authenticator` interface to create a local authentication handler.

About this task

Local authentication handlers can be implemented only in Java.

Procedure

1. Create a Java class that implements `Authenticator`.

```
private static class ExampleControlAuthenticationHandler
extends Stream.Default
implements ControlAuthenticator {

 private static final Map<String, byte[]> PASSWORDS = new
 HashMap<>();
 static {
  PASSWORDS.put("manager",
 "password".getBytes(Charset.forName("UTF-8")));
  PASSWORDS.put("guest",
 "asecret".getBytes(Charset.forName("UTF-8")));
  PASSWORDS.put("brian",
 "boru".getBytes(Charset.forName("UTF-8")));
  PASSWORDS.put("another",
 "apassword".getBytes(Charset.forName("UTF-8")));
  }

  @Override
  public void authenticate(
```

```
  String principal,
  Credentials credentials,
  Map<String, String> sessionProperties,
  Map<String, String> proposedProperties,
  Callback callback) {

   final byte[] passwordBytes = PASSWORDS.get(principal);

   // If the principal is in the table and has provided a valid
 password
   // then further processing of the properties may be applied
   if (passwordBytes != null &&
   credentials.getType() == Credentials.Type.PLAIN_PASSWORD &&
   Arrays.equals(credentials.toBytes(), passwordBytes)) {

    // The manager principal is allowed all proposed properties
    if ("manager".equals(principal)) {
    // manager allows all proposed properties
    callback.allow(proposedProperties);
    }
  // The principal brian is allowed all proposed properties and
 also
  // gets the 'super' role added
   else if ("brian".equals(principal)) {
    final Map<String, String> result =
    new HashMap<>(proposedProperties);
    final Set<String> roles =
    Diffusion.stringToRoles(
    sessionProperties.get(Session.ROLES));
    roles.add("super");
    result.put(Session.ROLES, Diffusion.rolesToString(roles));
    callback.allow(result);
    }
  // All other valid principals are allowed but with no proposed
  // properties assigned to the session
   else {
   callback.allow();
   }
  }
  // If the principal is not in the table it is denied access
   else {
   callback.deny();
   }
 }
}
```

a) Implement the `authenticate` method.
b) Use the `allow`, `deny`, or `abstain` method on the `Callback` object to respond with the authentication decision.

2. Package your compiled Java class in a JAR file and put the JAR file in the `ext` directory of your Diffusion installation.

   This includes the authentication handler on the server classpath.

3. Edit the `etc/Server.xml` configuration file to point to your authentication handler.

   Include the `authentication-handler` element in the list of authentication handlers. The order of the list defines the order in which the authentication handlers are called. The value of the `class` attribute is the fully qualified name of your authentication handler class. For example:

```
<security>
    <authentication-handlers>
```

```
        <authentication-handler
 class="com.example.ExampleAuthenticationHandler" />

    </authentication-handlers>
</security>
```

4. Start or restart the Diffusion server.
   - On UNIX-based systems, run the `diffusion.sh` command in the *`diffusion_installation_dir`*`/bin` directory.
   - On Windows systems, run the `diffusion.bat` command in the *`diffusion_installation_dir`*`\bin` directory.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

Authentication on page 149
You can implement and register handlers to authenticate clients when the clients try to perform operations that require authentication.

Related tasks

Developing a composite authentication handler

Developing a control authentication handler on page 352
Implement the `ControlAuthenticator` interface to create a control authentication handler.

Developing a composite control authentication handler

Using Maven to build Java Diffusion applications

Apache™ Maven is a popular Java build tool and is well supported by Java IDEs. You can use Apache Maven to build your Diffusion applications.

The Push Technology public Maven repository

Push Technology publishes Diffusion components and related artifacts to a public Maven repository at the following location: http://download.pushtechnology.com/maven.

The published artifacts include the following:

Table 39: Artifacts

Artifact	Maven coordinates	Description
Diffusion API	`com.pushtechnology.diffusion:diffusion-api:jar:6.7.0`	The Diffusion API interfaces only. Use this artifact for compilation only. The JAR includes the source and Javadoc attachments.
Diffusion Clients	`com.pushtechnology.diffusion:diffusion-client:jar:6.7.7`	The Diffusion client library.

To use the Push Technology public Maven repository, add the following repository description to your `pom.xml` file:

```
<repositories>
    <repository>
```

```
        <id>push-repository</id>
        <url>https://download.pushtechnology.com/maven/</url>
    </repository>
</repositories>
```

Build client applications

You can build and run Diffusion Java client applications without installing the Diffusion product. The Diffusion client JAR is all you need.

The following `pom.xml` shows how to declare the appropriate dependencies:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>myclient</artifactId>
    <version>1.0-SNAPSHOT</version>

    <repositories>
        <repository>
            <id>push-repository</id>
            <url>https://download.pushtechnology.com/maven/</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>com.pushtechnology.diffusion</groupId>
            <artifactId>diffusion-client</artifactId>
            <version>6.7.7</version>
        </dependency>
    </dependencies>
</project>
```

Build server application code with Maven

The Diffusion API for server application code is not available in the Push Technology public Maven repository. To build server components, you must install the product locally and depend on `diffusion.jar` using a Maven system scope.

The following `pom.xml` shows to declare the dependency on `diffusion.jar`. To use it, you must set the `DIFFUSION_HOME` environment variable to the absolute file path of your Diffusion installation.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.examplecorp</groupId>
    <artifactId>mypublisher</artifactId>
    <version>1.0-SNAPSHOT</version>
```

```
    <dependencyManagement>
        <dependency>
            <groupId>com.pushtechnology.diffusion</groupId>
            <artifactId>diffusion</artifactId>
            <version>local-installation</version>
        </dependency>
    </dependencyManagement>

    <profiles>
        <profile>
            <activation>
                <property>
                    <name>env.DIFFUSION_HOME</name>
                </property>
            </activation>

            <dependencyManagement>
                <dependencies>
                    <dependency>
                        <groupId>com.pushtechnology.diffusion</
groupId>
                        <artifactId>diffusion</artifactId>
                        <version>local-installation</version>
                        <scope>system</scope>
                        <systemPath>${DIFFUSION_HOME}/lib/
diffusion.jar</systemPath>
                    </dependency>
                </dependencies>
            </dependencyManagement>
        </profile>
    </profiles>
</project>
```

Testing

This section covers some aspects of testing a Diffusion system.

Benchmarking suite

A benchmarking suite for Diffusion is available on GitHub. You can use this suite to test the latency and throughput of publishers.

The benchmarking suite is available at the following location: https://github.com/pushtechnology/diffusion-benchmark-suite.

The benchmarking suite works on Linux only and requires the following software be installed on the system:

- Apache Ant™
- Java with JDK
- Diffusion server

For more information about using the benchmarking suite, see the readme file in the GitHub project.

Creating a remote server definition

To connect to a Diffusion server that is not in the same cluster, create a remote server definition. This can be used by a remote topic view.

A remote server definition provides the configuration to connect to a Diffusion server belonging to a different cluster. Each server in the local cluster will establish a session with each remote server.

Currently only remote topic views use a remote server definition.

Create a remote server definition

Required permissions: control_server

A client can create a new remote server definition with default connection options.

If a remote server with the same name already exists, an error is returned.

If a topic view that specifies the named remote server exists, the remote server connection is immediately established. However, if no topic views mention the remote server, a connection is not established when it is created.

You can use the `>checkRemoteServer` API method to check if the connection works.

Parameters:

name
: the name of the remote server

url
: the URL to use to connect to the primary server

principal
: the name of a principal used by the remote server to connect to the primary server.

credentials
: to use for connecting to the primary server

Java and Android

```
default CompletableFuture<RemoteServers.RemoteServer>
 createRemoteServer(String name,
    String url,
    String principal,
    Credentials credentials)
```

Removing a remote server definition

If you remove a remote server definition, any topic views depending on it will automatically deactivate.

Related concepts

Sharing data with remote servers on page 113
Remote topic views enable you to share topic data between servers that are not in the same cluster.

Part V

Administrator Guide

This guide describes how to deploy, configure, and manage your Diffusion solution.

In this section:

- Installing the Diffusion server
- Configuring your Diffusion server
- Starting the Diffusion server
- Network security
- Going to production
- Tuning
- Managing and monitoring your running Diffusion server
- Web servers
- Load balancers
- MQTT support
- Kafka adapter
- JMS adapter
- CDC database adapter
- Demos

Installing the Diffusion server

You can install the Diffusion server using the installer, through Docker, or through Red Hat Package Manager.

Review the system requirements before installing Diffusion.

Download Diffusion from the following location: https://www.pushtechnology.com/developers/releases/6.7

The Diffusion installation includes a developer license that allows up to five concurrent connections to the Diffusion server. To use Diffusion in production, you can obtain a production license from community.pushtechnology.com or by contacting our sales team at sales@pushtechnology.com .

System requirements for the Diffusion server

Review this information before installing the Diffusion server.

The Diffusion server is certified on the system specifications listed here. In addition, the Diffusion server is supported on a further range of systems.

Certification

Push Technology classes a system as certified if the Diffusion server is fully functionally tested on that system.

We recommend that you use certified hardware, virtual machines, operating systems, and other software when setting up your Diffusion servers.

Support

In addition, Push Technology supports other systems that have not been certified.

Other hardware and virtualized systems are supported, but the performance of these systems can vary.

More recent versions of software and operating systems than those we certify are supported.

However, Push Technology can agree to support Diffusion on other systems. For more information, contact Push Technology.

Physical system

The Diffusion server is certified on the following physical system specification:

- Intel Xeon E-Series Processors
- 8 Gb RAM
- 8 CPUs
- 10 Gigabit NIC

Network, CPU, and RAM (in decreasing order of importance) are the components that have the biggest impact on performance. High performance file system and disk are required. Intel hardware is used because of its ubiquity in the marketplace and proven reliability.

Virtualized system

The Diffusion server is certified on the following virtualized system specification:

Host

- Intel Xeon E-Series Processors
- 32 Gb RAM
- VMware vSphere 5.5

Virtual machine

- 8 VCPUs
- 8 Gb RAM

When running on a virtualized system, over-committing VCPUs (assigning too many VCPUs compared to the processors available on the host) can cause increased latency and unpredictable performance. Consult the VMWare Performance Best Practices documentation for details.

Operating system

Diffusion is certified on the following operating systems:

- Red Hat7.2+
- Windows Server 2012 R2 and 2016

We recommend you install your Diffusion server on a Linux-based operating system with enterprise-level support available, such as Red Hat Enterprise Linux.

Operating system configuration

If you install your Diffusion server on a Linux-based operating system and do SSL offloading of secure client connections at the Diffusion server, you must disable transparent huge pages.

If you install your Diffusion server on a Linux-based operating system but do not do SSL offloading of secure client connections at the Diffusion server, disabling transparent huge pages is still recommended.

Having transparent huge pages enabled on the system your Diffusion server runs on can cause extremely long pauses for garbage collection. For more information, see https://access.redhat.com/solutions/46111.

Java

The Diffusion server is supported on any JVM that meets the following requirements:

- The JVM has either passed the Java TCK or is an official AdoptOpenJDK build.
- The JVM version is Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later). We recommend that you use the latest available minor release of the JVM, regularly review and update the JVM as new minor releases become available, and prefer a distribution that provides regular updates.
- The JVM is HotSpot based.
- The target platform is Linux, macOS, or Windows.

There is a wide variety of free and commercial JDK distributions satisfying these requirements including:

- Oracle JDK (https://www.oracle.com/technetwork/java/javase/downloads/index.html). Production use requires a commercial licence from Oracle.
- AdoptOpenJDK (https://adoptopenjdk.net/).
- Amazon Corretto (https://aws.amazon.com/corretto/).
- Azul Zulu (https://www.azul.com/downloads/zulu/).
- Red Hat OpenJDK (https://developers.redhat.com/products/openjdk/overview/).

We recommend against using Oracle OpenJDK (http://jdk.java.net/). Oracle OpenJDK is a reference implementation, and release updates are provided for a limited period, typically only six months.

Diffusion 6.7 is certified against the following JVMs:

- AdoptOpenJDK 8u212
- AdoptOpenJDK 11.0.3+7
- Oracle JDK 8u131

Reduced footprint JVM distributions

Some distributions have optional packages that are minimal versions of the JVM and omit certain components, either to reduce the size of the JVM (and so the required disk space and time to download), or as part of security hardening. Additionally, the end user may remove unnecessary components from a JDK. JDKs up to version 8 had a standard reduced footprint repackaging called the "Java Runtime Environment (JRE)", but this name has now been dropped to allow distributions or the end user to choose from a variety of different repackaging strategies.

Examples of components that are frequently removed include:

- Tools that are only required by developers (e.g. the javac compiler; Java Mission Control; jconsole). This was the primary focus of the JRE.
- Graphical user interface libraries. Packages without these components are typically referred to as "headless".

Diffusion does not require developer tools nor graphical user interface libraries. Although Push Technology Ltd supports the use of Diffusion with a reduced footprint JVM, we recommend that a full JDK installation is used. This is because Java's diagnostic tools, in particular Java Mission Control, have proven to be particularly useful as part of performance tuning and problem diagnosis.

JVM configuration

If you do SSL offloading of secure client connections at the Diffusion server, you must ensure that you constrain the maximum heap size and the maximum direct memory size so that together these to values do not use more than 80% of your system's RAM.

Networking

Push Technology recommends the following network configurations:

- 10 Gigabit network
- Load balancers with SSL offloading
- In virtualized environments, enable SR-IOV.

  For more information about how to enable SR-IOV, see the documentation provided by your virtual server provider. SR-IOV might be packaged using a vendor-specific name.

Validation script

You can run an environment validation script to detect problems with the install environment.

The script uses the jjs command line tool which is included with Java 8 and above.

1. In the directory where you installed Diffusion, navigate to `/bin`.
2. Run the environmentValidation.js script:
   - On Linux, run `./environmentValidation.js`
   - On Windows, make sure `<java_home>` is added to your path, then run: `jjs environmentValidation.js`

Note that macOS is not certified as a production system for Diffusion, so the validation script does not support macOS. You can still install and run Diffusion on macOS for development purposes.

Client requirements

For information about the supported client platforms, see Platform support for the Diffusion API libraries on page 30.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Using the Diffusion server installer in graphical mode

You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Before you begin

You must have Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later) installed on your system to run the installer.

The following assumes you are using a graphical display to access the computer to which you want to install Diffusion. For installation from a non-graphical command line, see Using the Diffusion server installer in headless mode on page 403.

About this task

To install Diffusion, complete the following steps:

Procedure

1. Open the Diffusion download page at https://www.pushtechnology.com/developers/releases/6.7 in a web browser. Click on the **Download Diffusion** link and download the `diffusion-installer-6.7.7.jar` file to a temporary directory.
2. In the temporary directory, double-click the `diffusion-installer-6.7.7.jar` file.

   Depending on the installation platform, you may first need to set the file to be executable, and you may need to accept warnings about running a file downloaded from the Internet.

The graphical installer launches.

3. At the **Location** step, select the install destination.

   Use a new directory for each installation. Click **Next**.

4. At the **Components** step, select the components you want to install

   Click **Next**.

5. Diffusion will be installed. Click **Next**.

   Review the information presented in the **Summary** page. It contains instructions on how to start the Diffusion server, and how to install a different license file.

   Click **Done** to exit the graphical installer.

Results

You have successfully downloaded and installed Diffusion.

What to do next

Next:

- Edit the configuration of your Diffusion server to suit your requirements. For more information, see Configuring your Diffusion server on page 414.
- Start your Diffusion server using the start up scripts located in the `bin` directory of your Diffusion installation.

  If you are using Java 8, run the `diffusion.bat` file, if on Windows, or the `diffusion.sh` file, if on Linux or OS X/macOS.

  If you are using Java 11, run the `diffusion_java11.bat` file, if on Windows, or the `diffusion_java11.sh` file, if on Linux or OS X/macOS.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27

Review this information before installing the Diffusion server.

Using the Diffusion server installer in headless mode

You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Before you begin

You must have Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later) installed on your system to run the installer.

The following assumes you are using a non-graphical terminal to access the computer to which you want to install Diffusion. For installation from a graphical display, see Installing the Diffusion server on page 398.

About this task

You can install in headless mode in circumstances where the graphical installer cannot be used or is not appropriate.

Procedure

1. Open the Diffusion download page at https://www.pushtechnology.com/developers/releases/6.7 in a web browser. Click on the **Download Diffusion** link and download the `diffusion-installer-6.7.7.jar` file to a temporary directory.
2. In the terminal window, change to the temporary directory and type the following command:

   ```
   java -jar diffusion-installer-6.7.7.jar -console
   ```
3. Enter the full path to the directory in which to install Diffusion, or select the default.
4. Follow the prompts to select the optional components to install.

Results

You have successfully downloaded and installed Diffusion.

What to do next

Next:

- Edit the configuration of your Diffusion server to suit your requirements. For more information, see Configuring your Diffusion server on page 414.
- Start your Diffusion server using the start up scripts located in the `bin` directory of your Diffusion installation.

  If you are using Java 8, run the `diffusion.bat` file, if on Windows, or the `diffusion.sh` file, if on Linux or OS X/macOS.

  If you are using Java 11, run the `diffusion_java11.bat` file, if on Windows, or the `diffusion_java11.sh` file, if on Linux or OS X/macOS.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410

After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27
Review this information before installing the Diffusion server.

Installing the Diffusion server using Red Hat Package Manager

Diffusion is available as an RPM file from the Push Technology website.

About this task

On Linux systems that have Red Hat Package Manager installed, you can use it to install Diffusion.

Procedure

1. Go to the Diffusion download page:
   https://www.pushtechnology.com/developers/releases/6.7
2. Click on the following download link to download the required RPM file:
   - Diffusion RPM (`diffusion-n.n.n_build.noarch.rpm`)
3. Copy this file to a temporary directory on the system where Diffusion is to be installed.
4. In the terminal window, change to the directory where the Diffusion RPM file is located.
5. Type the following command:

   ```
   rpm -ivh diffusion-n.n.n_build.noarch.rpm
   ```

   where *n.n.n* is the Diffusion release number and *build* is an additional string containing numbers to represent the build level.

Results

Diffusion is installed in the following directory: `/opt/Diffusion`. A startup script is installed in the `/etc/init.d` directory that enables Diffusion to start when you start the system.

What to do next

Your Diffusion installation includes a development license that allows connections from up to five clients. To use Diffusion in production, you can obtain a production license from .

Copy the license file into the `/etc` directory of your Diffusion installation.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27
Review this information before installing the Diffusion server.

Installing the Diffusion server using Docker

Diffusion is available as a Docker® image from Docker Hub.

Before you begin

You must have Docker installed on your system to run Diffusion from a Docker image. For more information, see https://docs.docker.com/userguide/ .

About this task

You can use Docker to install the Diffusion server, and a minimal complete set of its dependencies, on a Linux system. This image contains a Diffusion server with a trial license and default configuration and security.

Using Docker enables you to install the Diffusion server in an isolated and reproducible way.

Procedure

1. Pull the latest version of the Diffusion image.

   ```
   docker pull pushtechnology/docker-diffusion:6.7.7
   ```

   If you receive an error about the license file, check you are pulling the latest version available.
2. Run the image.

   ```
   docker run -p 8080:8080 image_id
   ```

Where *image_id* is the ID of the image to run. You can find the image ID using

```
docker images
```

Port 8080 is the port that is configured to allow client connections by default.

Results

Diffusion is now running in a container on your system. Clients can connect through port 8080.

> **Note:** This Diffusion instance contains well known security principals and credentials. Do not use it in production without changing these values.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27
Review this information before installing the Diffusion server.

Next steps with Docker

The Diffusion image on Docker Hub includes the default configuration, default security, and trial license. Additional steps are required to secure and configure the Diffusion server.

Procedure

1. Create a Dockerfile that contains commands to configure a Diffusion image for your use.
   a) Base your Docker image on the Diffusion image.

   ```
   FROM pushtechnology/docker-diffusion:6.7.7
   ```

   b) To use Diffusion in production, obtain a production license from Push Technology.

   The default Diffusion image includes a development license that allows connections from up to five clients.

c) Copy the production license into the `/opt/diffusion/etc` directory of your Diffusion image.

```
ADD license_file /opt/diffusion/etc/license.lic
```

Where *license_file* is the path to the production license relative to the location of the Dockerfile.

d) Create versions of the Diffusion configuration files that define your required configuration.

For more information, see Configuring your Diffusion server on page 414.

e) Copy these configuration files into the `/opt/diffusion/etc` directory of your Diffusion image.

```
ADD configuration_file /opt/diffusion/etc/file_name
```

Where *configuration_file* is the path to the configuration file relative to the location of the Dockerfile and *file_name* is the name of the configuration file.

f) Create versions of the `Security.store` and `SystemAuthentication.store` that define roles, principals and authentication actions for your security configuration.

For more information, see Pre-defined roles on page 147, DSL syntax: security store on page 368, and DSL syntax: system authentication store on page 356.

You can instead choose to edit these files using a Diffusion client. However, your Diffusion server is not secure for production use until you do so.

g) Copy these store files into the `/opt/diffusion/etc` directory of your Diffusion image.

```
ADD store_file /opt/diffusion/etc/file_name
```

Where *store_file* is the path to the store file relative to the location of the Dockerfile and *file_name* is the name of the store file.

h) Include any additional configuration actions you want to perform on your image in the Dockerfile.

2. Build your image.

Run the following command in the directory where your Dockerfile is located:

```
docker build .
```

Results

The image you created contains a configured Diffusion server ready for you to use in your solution. You can run multiple identically configured Diffusion servers from this image.

The Diffusion license

Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

The restricted default license is not for use in production.

If you want a more capable license, visit community.pushtechnology.com.

From there you can get:

- a free Community Production license which allows more connections than the default license; suitable for smaller scale projects using a single server in production, but does not allow use of premium features for high availability and scaling
- a free Community Evaluation license which allows use of premium features for high availability and scaling (time-limited)

Contact our sales team to discuss a commercial production license or an extended evaluation license.

Related concepts

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27
Review this information before installing the Diffusion server.

License restrictions

The Diffusion license can include restrictions on how the Diffusion server is used.

Environments

A Production license must not be used on a Development server, and a Development license must not be used on a Production server. Order separate licenses defined as Production, QA/Testing, Disaster Recovery, and Development.

License expiry

All license files provided by Push Technology include an expiry date. To continue to use Diffusion after this date you must replace your license file with an updated license file.

The Diffusion server logs the number of days remaining on your license every day at midnight and when the server starts (PUSH-000202 on page 559).

When the license has expired, the Diffusion server stops working within 24 hours. A message is logged when the license expires (PUSH-000203 on page 560).

Concurrent client connections

An instance of the Diffusion server is licensed to only allow up to a certain number of client connections at the same time.

A license can include a soft limit and a hard limit on concurrent client connections. When the soft limit is reached, the Diffusion server logs a message (PUSH-000201 on page 559) to say that the soft limit has been reached. When the hard limit is exceeded, the Diffusion server rejects connections and logs a message (PUSH-000056 on page 550) to say that the hard limit has been reached.

Connection limit pooling

If you have a license that enables topic and session replication, the soft and hard limits are pooled between servers.

For example, you have a cluster of three servers, each with a soft limit of 5 and a hard limit of 10. The cluster-wide limits are the total of each individual server's limits, giving a cluster-wide soft limit of 15 (3x5) and a hard limit of 30 (3x10).

Any new client connections are checked against the cluster-wide limits. A server can exceed the hard limit of its individual license, provided that the cluster-wide limit is not exceeded. For example, one of the servers in our example above could have more than 10 client connections, provided the total number of connections to the cluster did not exceed 30.

Fan-out limit pooling

If you have a license that enables fan-out, the hard limit for fan-out connections is pooled between servers in a cluster.

Total number of topics

A Diffusion license can specify a maximum total number of topics. If the number of topics is exceeded, the server logs a message (PUSH-000711 on page 595).

MAC addresses or IP addresses

An instance of the Diffusion server can be licensed to run only on systems with a certain range of IP addresses or MAC addresses.

On startup, the Diffusion server checks the IP address or MAC address of the system the server runs on. If the Diffusion server cannot read the IP or MAC address of the host system, it logs a message (or) and does not start. If the IP or MAC address of the host system is not in the licensed address range, the server logs a message (or) and does not start.

CPU cores

An instance of the Diffusion server can be licensed to run only on systems with a certain number of CPU cores.

On startup, the Diffusion server checks the number of CPU cores available to the JVM at runtime.

Diffusion version

A Diffusion license can be valid for specific versions of Diffusion only.

If you use a license file with a version of Diffusion that it is not valid for, the Diffusion server logs a message (PUSH-000199 on page 559) and does not start.

Updating your license file

You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

Before you begin

Obtain a new or renewed license file from Push Technology.

About this task

When your license file expires, the Diffusion server continues to run for another day before it stops. We recommend you update your license file before your existing license file expires.

Procedure

1. Copy the new license file (`license.lic`) over the existing file in the *`diffusion_directory`*`/etc` directory.

   You do not need to stop or restart the server.

   Note that in versions before Diffusion 6.6, the license filename was `licence.lic`. Versions 6.6 and above will look for a license file called `license.lic` first, and if that file does not exist, `licence.lic` will be used if it is present.
2. Check that the timestamp of `license.lic` has updated.
   - On Windows, you might have to use the following command to copy the file over and force the timestamp to update: `COPY /B license.lic +,,`
3. Diffusion checks the timestamp of the `license.lic` every minute. If the license file has been updated, Diffusion reloads it and logs this to stdout.
4. You can verify that the license file has been updated in the server by accessing the mbean **com.pushtechnology.diffusion** > **Server** > **LicenseExpiryDate**

Installed files

After installing Diffusion the following directory structure exists:

Table 40: Installed files

Folder name	Contents
`adapters`	Adapters to connect Diffusion to other messaging systems: • JMS • Push Notifications
`bin`	Executables for starting Diffusion
`clients`	Client Diffusion API libraries and related artifacts for all supported platforms.
`data`	Files used by the console, and third-party components. This directory is always on the server classpath.
`demos`	The compiled DAR files and source code for the demos issued with Diffusion. For more information, see Demos on page 644.
`deploy`	If you selected during the install process to deploy the demos, the demo DAR files are in this directory.
`docs`	License information, release notes, and install notes.
`etc`	Diffusion initial configuration files and store files for security configuration. The `Security.store` and `SystemAuthentication.store` are not the working configuration files, which are stored in the `persistence` directory. However, if the server has never been started, there are no files in `persistence`, and the files in this directory are copied into `persistence` on first startup.

Folder name	Contents
	For more information, see Configuring your Diffusion server on page 414.
`examples`	Example code that uses the Diffusion APIs.
`ext`	This directory, together with any jar files in this directory or subdirectories, are available through the classloader used to deploy application code to the Diffusion server. You can add library jar files to this directory that are required by application code such as local authentication handlers.
`html`	Files that are used by the default web server for issuables accessible through the browser.
`lib`	The main Diffusion server JAR file, third-party libraries, and additional server-side components.
`logs`	The directory to which Diffusion server and web server logs are written.
`tools`	Tools and utilities that help with testing and deploying Diffusion. For more information, see Tools and utilities on page 411
`xsd`	The schema files for the XML configuration files used by the server.

Tools and utilities

The following table describes the some of the contents of the `tools` directory.

> **Note:** The files present and their suffixes vary according to the platform that the product is installed on.

Table 41: Tools and utilities

Tool	Description
`/systemd`	Sample `systemd` files to start the Diffusion server as daemon on Linux systems.
`war.xml`	Example `war.xml` file
`web.xml` and `sun-web.xml`	Example `web.xml` files

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404

Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Verifying the Diffusion installation on page 412
Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

Related reference

System requirements for the Diffusion server on page 27
Review this information before installing the Diffusion server.

Verifying the Diffusion installation

Start your Diffusion server, review the logs, and connect to the console to verify that your installation is correct.

About this task

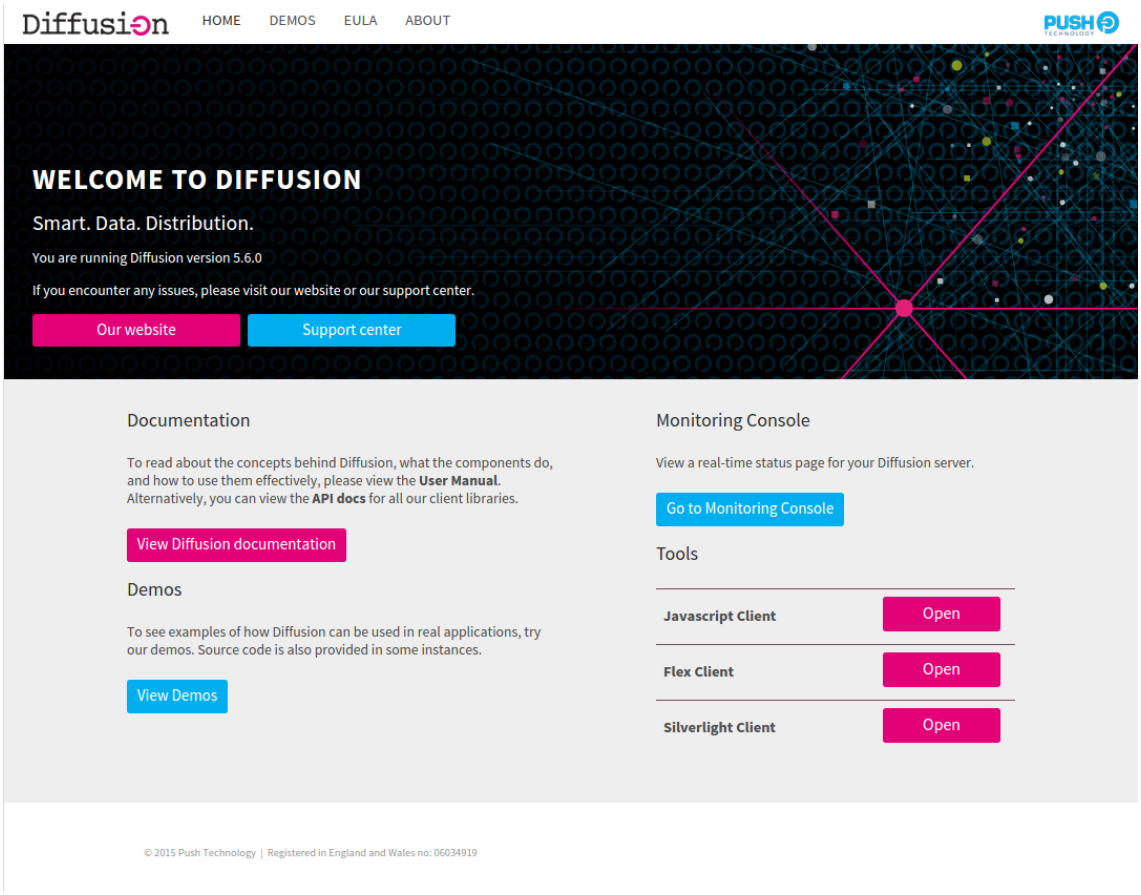
After installation, all of the Diffusion files are available in the directory specified during installation.

Procedure

1. Start the Diffusion server using one of the start script located in the `bin` directory of your Diffusion installation.
   - On Windows, use the `diffusion.bat` file.
   - On Linux, macOS, or UNIX, use the `diffusion.sh` file.
2. Inspect the log messages to ensure that the Diffusion server started successfully.

   The terminal window displays logging information about the status of the Diffusion server. A log message containing the following text indicates that the server started successfully: `INFO|main|PUSH0165|Diffusion Server started.|com.pushtechnology.diffusion.DiffusionController` This line is typically the last one to be printed on terminal.
3. Inspect all log messages displayed in the terminal to search for `WARN` messages to ensure that all components have started correctly.
4. Open a browser and navigate to `http://`*`serverAddress`*`:8080` (or `http://localhost:8080`)

   The browser shows the Diffusion landing page.



The landing page provides links to information regarding legal terms and conditions (for example, EULA), user guides, API documentation and demos.

The Diffusion server is ready to be used.

5. If you chose to install the demos, you can access them from the landing page.

Related concepts

The Diffusion license on page 407
Diffusion includes a restricted default license that enables you have up to 5 concurrent sessions connected to the Diffusion server.

Installed files on page 410
After installing Diffusion the following directory structure exists:

Related tasks

Using the Diffusion server installer in graphical mode on page 401
You can install Diffusion by running the installer. The Diffusion installer is available from the Push Technology website.

Using the Diffusion server installer in headless mode on page 403
You can install Diffusion from the command line by running the installer in headless mode. The Diffusion installer is available from the Push Technology website.

Installing the Diffusion server using Red Hat Package Manager on page 404
Diffusion is available as an RPM file from the Push Technology website.

Installing the Diffusion server using Docker on page 405
Diffusion is available as a Docker® image from Docker Hub.

Related reference

System requirements for the Diffusion server on page 27

Review this information before installing the Diffusion server.

Configuring your Diffusion server

You can configure the Diffusion server using XML files which normally reside in the `etc` directory. You can also configure user security on the Diffusion server using the `.store` files in the `etc` directory.

Alternatively, a Diffusion server can be instantiated in a Java application and configured programmatically.

In a Java client environment certain properties can also be configured programmatically.

All properties (whether configured from XML or programmatically) are available to read programmatically from within the Java API.

XML configuration

Configuring a Diffusion server using XML property files

XML Property files

A Diffusion server is configured using a set of XML property files typically loaded from the `etc` folder. In a new Diffusion installation example versions of these files are provided which can be edited as required.

XML is used rather than standard property files due to the hierarchic nature and the ability to support repeating groups.

XSD files are issued that define the content of the XML property files and this section summarizes the XSD content.

Configuration path loading

You can pass a parameter to Diffusion upon startup so that files are not automatically loaded from the `etc` folder but loaded from a different folder. This folder does not have to contain the complete set of XML files, but the file is loaded from the specified folder first, if it exists. If it does not, Diffusion loads the configuration file from the `etc` folder. When Diffusion starts, it logs where each configuration file has been loaded from.

XML Value types

When XML values are loaded, the schema is checked so that we know that it is valid, but to aid configuration, there are some extra data types. When values are loaded, they are trimmed of leading and trailing white space.

Table 42: XML Value types

Data type	Meaning
push:boolean	true or false
push:string	String value
push:int	A number between -2,147,483,648 and 2,147,483,647

Data type	Meaning
push:long	A number between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807
push:double	An 8 byte IEEE 754 floating point number: from +/- 2^-1074 to +/- (2-(2^-52))·2^1023
push:port	A positive number but less than 65535
push:millis	A string that represents the number of milliseconds. Append the mnemonic for the time unit. The mnemonic can be either upper or lower case. **s** Seconds **m** Minutes **h** Hours **d** Days 360000, 360s, 6m all represent 6 minutes
push:bytes	A string that represents the number of bytes. Append the mnemonic size unit. The mnemonic can be either upper or lower case. **k** Kilobytes **m** Megabytes **g** Gigabytes 6291456, 6144k, 6m, all represent 6 Megabytes
push:log-level	A log level can be ERROR, WARN, INFO, DEBUG, or TRACE.
push:percent	A value that represents a percentage, this can have the trailing percent sign (%)
push:positiveNonZeroInt	A number between 1 and 2,147,483,647
push:positiveInt	A number between 0 and 2,147,483,647
push:positiveNonZeroLong	A number between 1 and 9,223,372,036,854,775,807
push:positiveLong	A number between 0 and 9,223,372,036,854,775,807
<element>	This notation is used to indicate a complex element type. It can also be List<element> to indicate a repeating property group.

Environmental values

When defining custom configurations, you can define environmental variables that can be reused in all XML property files. These variables can be defined in the `etc/Env.xml` property file to be used in all other property files. Suppose, for example, the `etc/Env.xml` file defines a `server-name` variable, with value `d-unit` as follows:

```
<env>
 <property name="server-name">d-unit</property>
</env>
```

The `server-name` variable can be used in all other property files, where the value `d-unit` is appropriate, either as a value for an attribute, as in

```
<server name={server-name}>
...
</server>
```

or as a name for an element as in:

```
<server>{server-name}</server>
```

As a side remark, it is worth noting that names can be combined to provide malleable environmental variables. Suppose for instance `Env.xml` contains the following entries:

```
<env>
 <property name="server-name">myServer</property>
 <property name ="server-version">V2.0</property>
</env>
```

Then `server-name` and `server-version` can be combined, for instance within the same `etc/Env.xml`, as

```
<property name="server-and-version">{server-name}-{server-version}</
property>
```

and used in all other configuration files.

Obfuscation tool

Use the obfuscation tool to protect sensitive strings such as passwords in configuration files.

Obfuscation tool

The Diffusion configuration files can contain sensitive data like:

- fan-out connection passwords
- keystore passwords

Use the obfuscation tool to make it harder for an attacker to read the passwords. The tool converts them to a form that the server can understand, but which is not easily readable by a casual observer.

The tool is a command-line script in the `bin` directory called `obfuscate.sh` (or `obfuscate.bat` for Windows).

The script takes strings representing the passwords or other values you want to protect as command line arguments.

It writes out the obfuscated version of each argument in order.

Copy the output and use it in the Diffusion configuration file in place of the original string.

> **Note:** The obfuscation method provides superficial protection against casual browsers. To provide better protection, ensure the file can only be read by trusted users.

Programmatic configuration

An alternative to configuring a Diffusion server using XML property files is to instantiate a Diffusion server within a Java application and configure it with the Server API before starting it.

If desired, some properties can be loaded from XML files and some supplied programmatically or default properties can be bootstrapped from XML files and overridden programmatically before the server is started.

Most server properties can be configured only before the server is started. Instantiate the server within an application and configure before starting the server. However, certain configuration items can be configured at any time during the life of the server. The API documentation makes it clear if a property can be changed at runtime.

Because the properties that can be set programmatically reflect those that can be set in XML this section does not describe the properties in detail. The XSD property descriptions or the Server API documentation can be consulted for full details.

As well as allowing configuration properties to be set the configuration API also allows all properties that can be configured to be read at runtime.

Using the Server API for configuration

Server API documentation

Server API documentation for full details of the API.

General use

The Server API only affects server-side configuration.

From within server-side code, the server configuration root can be obtained using `ConfigManager.getServerConfig()` which exposes all of the server side configuration also.

From the configuration root you can navigate to any subordinate configuration objects to view them or set their properties.

Most properties cannot be changed after the server has started and they become locked so any attempt to change them results in an exception. Certain properties (such as conflation and connection policies) can be changed at runtime. The API documentation makes it clear which properties can be changed at runtime.

For configuration objects which are optional but there can be many (multiplicity 0..n), there are appropriate add methods to add new objects.

In these cases there are also methods to obtain the full list or to obtain a specific one by name. In many cases there are also methods to remove an object.

> **Note:** When there must be at least one object (multiplicity 1..n), you must configure at least one. However, if a server is started with missing configuration of this kind, suitable defaults are normally created and a warning logged.

Single instance configuration objects (multiplicity 1..1) subordinate to the root can be obtained so that their properties can be changed (or read). So, for example the `Queues` object (an instance of `QueuesConfig`) can be obtained using the `getQueues()` method.

When a single configuration object is optional (multiplicity 0..1), the `get` method can return null if it has not been defined. In this case to set it the set method (as opposed to add) returns the object created. An example of this is the file service (`FileServiceConfig`) on a web server (`WebServerConfig`) as shown in the following example code:

```
ServerConfig config = ConfigManager.getServerConfig();
WebServerConfig webServer = config.addWebServer("MyWebServer");
FileServiceConfig fileService = webServer.setFileService("File
 Service");
```

Configuring a server

After instantiating a Diffusion server in Java the root of the server configuration tree can be obtained from the server object itself and configuration objects can be navigated to and changed as required before starting the server.

For example, the following code shows how to add a connector that accepts client connections on port 9090:

```
DiffusionServer server = new DiffusionServer();
ServerConfig config = server.getConfig();
ConnectorConfig connector = config.addConnector("Client Connector");
connector.setPort(9090);
connector.setType(Type.CLIENT);
server.start();
```

In reality, it is best to configure far more values. However, if any essential objects are omitted (such as queues), suitable defaults are created when the server starts and a warning is logged.

Configuring the Diffusion server

Use the `Server.xml` configuration file to configure the core behaviors of the Diffusion server.

Configuring fan-out

Configure the the Diffusion server to act as a client to one or more other Diffusion servers and replicate topics from those servers.

Use the `fanout` section of the `Server.xml` configuration files to define client connections for this secondary server to make to one or more primary servers and the topics on those primary servers to replicate locally.

Each `fanout-connection` element represents a client connection that your Diffusion server makes to another Diffusion server in your solution.

```
<fanout>
    <connection>
        <url>ws://primary_server_hostname:8080</url>
        <principal>client</principal>
        <password>password</password>
        <retry-delay>1000</retry-delay>
        <reconnect-timeout>60s</reconnect-timeout>
        <recovery-buffer-size>1024</recovery-buffer-size>
        <input-buffer-size>1024k</input-buffer-size>
        <output-buffer-size>1024k</output-buffer-size>
        <link><selector>?topic_path//</selector></link>
    </connection>
</fanout>
```

Connection

Use the `url` element to specify the URL of the primary server and the transport and port used for the connection.

Permissions

When connecting to another Diffusion server as a client, this secondary server can provide a principal and associated password. If a principal is not provided, the secondary server connects anonymously

To subscribe to topics on the primary server and replicate them locally, the secondary server's client session must have the select_topic and read_topic permissions for those topics. Ensure that the principal this secondary server uses is assigned a role with the appropriate permissions on the primary server. If the secondary server connects anonymously to the primary server, ensure anonymous sessions on the primary server are assigned the appropriate permissions.

Reconnection

Use the `retry-delay` element to specify the time in milliseconds between the connection or reconnection attempts that the secondary server makes to the primary server.

Use the `reconnect-timeout` element to specify the maximum time in milliseconds that the secondary server will attempt to reconnect to its existing session on the primary server after a disconnection. If this element is not specified, a value of 0 is assumed and reconnection is not attempted.

If the secondary server is configured to attempt to reconnect, it keeps a buffer of messages sent to the primary server. Use the `recovery-buffer-size` element to configure the size of this buffer.

Replicating topics

Each `fanout-connection` has one or more `link` elements. Each `link` element uses a topic selector to specify a set of topics on the primary server to replicate on this secondary server.

> **Note:** The set of topics specified by a link cannot overlap the set of topics specified by any other link within either this `fanout-connection` or any of the others.

If you want missing topic handlers registered on the primary server to receive missing topic notifications when a subscription or fetch request is made on the secondary server to a part of the topic tree that matches a link selector, consider the following when configuring your secondary server links:

- Avoid using regular expressions in the selectors you use to configure when setting up fan-out links on the secondary server. Topic selectors containing regular expressions increase the likelihood of false negatives and false positives when propagating missing topic notifications.
- Ensure that the principal that the secondary server uses to make the fan-out connection to the primary server has the SELECT_TOPIC permission for the path prefix of the selector that triggered the missing topic notification.

For more information, see Using missing topic notifications with fan-out on page 118.

Configuring your primary server

The primary server in a fan-out configuration must be configured to handle serving the topics replicated by fan-out to the secondary server or servers.

Ensure that the primary server connector that the secondary server or servers connect to has a large enough queue to handle the number of primary server topics that will be replicated by fan-out. In the `Connectors.xml` file, inside the `<connector>` element that defines the connector used for fan-out connections, set the queue depth to greater than the number of fanned out topics:

```
<queue-definition>depth</queue-definition>
```

To allow the secondary server to reconnect, enable reconnection on the connector that the primary server uses to accept connections from the secondary server or servers. Ensure that the reconnection timeout (`keep-alive`) value for the connector is long enough to allow the secondary server time to reconnect. Set the maximum queue depth and recovery buffer sizes to values that are appropriate to the volume of messages you expect to occur between the primary and secondary servers.

For more information, see Connectors.xml on page 438.

Related concepts

Fan-out on page 116
Fan-out is a way to replicate topic information from primary servers on one or more secondary servers.

Configuring authentication handlers

Authentication handlers and the order that the Diffusion server calls them in are configured in the `Server.xml` configuration file.

To configure authentication handlers for your server, edit the `Server.xml` configuration file to include the following elements:

```
<security>
 <authentication-handlers>
  <authentication-handler class="com.example.LocalLDAPHandler" />
  <system-authentication-handler/>
  <control-authentication-handler handler-name="RemoteHandler" />
 </authentication-handlers>
</security>
```

Ordering your configuration handlers

The order of handler elements within the `<authentication-handlers>` element defines the order in which the authentication handlers are called. In the preceding example, `localLDAPHandler` is called first. If `localLDAPHandler` returns an ABSTAIN result, the system authentication handler is called next. If the system authentication handler returns an ABSTAIN result, `RemoteHandler` is called next.

Order your authentication handlers from least to most restrictive and configure your handlers to abstain unless they are to explicitly allow or deny the authentication request.

For more information, see Authentication on page 149.

Configuring local authentication handlers

Configure local authentication handlers by using the `<authentication-handler/>` element. The value of the attribute `class` is the class name for the handler.

You can configure any number of distinct local authentication handlers in the `Server.xml` file.

Configuring the system authentication handler

You can configure Diffusion to use the system authentication handler by using the `<system-authentication-handler/>` element. The system authentication handler uses information in the system authentication store to make authentication decisions.

You can configure the system authentication handler to be called at most once. This restriction is not enforced by the XSD for the `Server.xml` file, but the Diffusion server does enforce this restriction on the configuration.

Configuring control authentication handlers

Configure control authentication handlers are configured by using the `<control-authentication-handler/>` element. The value of the attribute `handler-name` is the name by which the handler was registered by the control client. Control clients use the AuthenticationControl feature to register the handler and passing the binding name as a parameter.

If no control client has registered a control authentication handler with the name defined in the configuration file, the response for that handler is ABSTAIN.

If you are using a cluster of Diffusion servers and configuration replication is enabled, authentication requests will be routed to across the cluster, meaning it is only necessary for a handler to connect to one server in the cluster.

Multiple control clients can register a control authentication handler with the same name. Registering a control authentication handler from multiple clients gives the following advantages:

- If one of the control clients becomes unavailable, another can handle the authentication request.
- Control clients can be changed or updated without affecting the authentication behavior.
- Authentication requests can be load balanced between the control clients.

You can configure any number of distinct control authentication handlers in the `Server.xml` file.

If you are using a cluster of servers, the security configuration must be the same on all the servers in the cluster.

> **Note:** To register a control authentication handler, an authenticating client must first connect to and authenticate with the server. We recommend that you configure a local authentication handler or the system authentication handler in the `Server.xml` file to authenticate the control client.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

Local authentication handlers on page 391
You can implement authentication handlers that authenticate client connections to the Diffusion server.

Example: Register an authentication handler on page 343
The following examples use the Diffusion API to register a control authentication handler with the Diffusion server. The examples also include a simple or empty authentication handler.

Authenticating new sessions on page 342
A client session can use the AuthenticationControl feature to authenticate other client sessions.

Related reference

System authentication handler on page 154

Diffusion provides an authentication handler that uses principal, credential, and roles information stored in the Diffusion server to make its authentication decision.

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

Configuring performance

Use the `Server.xml` configuration file to configure behaviors and parameters that affect the performance of the Diffusion server.

For more information on the factors to consider when configuring the performance of your Diffusion server, see the Tuning on page 497 section of this guide.

Configuring topic persistence

Use the `Server.xml` configuration file to configure the topic persistence feature.

Procedure

1. Edit this section of the `etc/Server.xml` file to configure persistence.

```
<persistence enabled="false">
                <!--
                <store-directory>dirName</store-directory>
                -->
    </persistence>
```

   - In the `persistence` element, set the `enabled` property to `true` to enable persistence. Removing the `enabled` property will also enable persistence.
   - By default, log files will be created in a directory called `persistence` in the Diffusion installation directory. If you want to use a different directory, uncomment the `store-directory` element and enter the full path to the directory where you want the files to be stored.
2. Restart the Diffusion server to load the configuration.

What to do next

Enabling persistence creates log files which can use a significant amount of file storage. Make sure to monitor the amount of space available in the server file system. See Topic persistence on page 120 for information about the approximate storage requirements.

If you want to back up or restore the persistence log, you should stop the Diffusion server.

When enabled, persistence is applied to all topics by default. You can disable persistence for an individual topic using the PERSISTENT topic property.

Related concepts

Topic persistence on page 120

Consider if you want to enable topic persistence for fast recovery of topic state when Diffusion servers restart.

Server.xml

This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

server

All server properties

The following table lists the elements that an element of type `server` can contain:

Name	Type	Description	Min occurs	Max occurs
server-name	push:string	The server name is used to identify this server if running in a cluster. If not specified, the local hostname is used.	0	1
persistence-directory	push:string	The path of a directory that is to be used to hold persistent files. This may be an absolute path or a relative path. A relative path will be taken to be relative to the Diffusion home directory. If this is not specified then the value of the persistence store-directory will be considered and if no directory is specified there then a relative path of 'persistence' will be used. The directory will be created if it does not already exist.	0	1
max-message-size	push:bytes	The maximum message size in bytes. This defines the maximum message size (including headers) that can be received.	1	1
multiplexer	multiplexer	Properties that define the multiplexers.	0	1
security	security	Properties relating to security (optional).	0	1
client-queues	client-queues	Definitions of client queues.	1	1
connection-timeouts	connection-timeouts	Timeout values relating to connections. If a value is not specified, defaults are used.	0	1
date-formats	date-formats	Date and time formats. DEPRECATED: since 6.5. This configuration is no longer used and will be removed in a future release.	0	1
thread-pools	thread-pools	Definitions of thread pools	1	1
selector-thread-pools	selector-thread-pools	Definitions of thread pools	0	1

Name	Type	Description	Min occurs	Max occurs
whois	whois	Definition of the WhoIs lookup service. If a value is not specified, no WhoIs service runs.	0	1
auto-deployment	auto-deployment	Automatic deployment properties (optional). If not specified then auto deployment is not enabled.	0	1
geo-ip	geo-ip	Properties relating to the Geo IP lookup facility. If a value is not specified, defaults are used.	0	1
usr-lib	usr-lib	User libraries (optional).	0	1
hooks	hooks	User hooks used in the server (optional)	0	1
fanout	fanout	Properties relating to fan-out (optional). If not specified then the server will not be enabled as a fan-out secondary server.	0	1
persistence	persistence	Properties relating to topic tree persistence (optional). If not specified, persistence will not be enabled.	0	1

multiplexer

Multiplexer configuration. Multiplexers are responsible for subscription evaluation and output processing. Each session hosted by the server is allocated to a multiplexer. Each multiplexer uses a CPU core.

The following table lists the elements that an element of type `multiplexer` can contain:

Name	Type	Description	Min occurs	Max occurs
size	push:positiveInt	The number of multiplexer instances. Each multiplexer uses a CPU core. If the server will host a large number of sessions, and there are spare CPU cores available, increase this number. If a value is not specified, a default value equal to half the number of available CPU cores is used.	0	1
latency-warning	push:millis	The multiplexer latency warning threshold. This setting controls the threshold at which to issue a warning if the multiplexer is taking too long to complete an operational cycle. The default value is 1000 (1 second). Warnings are logged to the server log at info level.	0	1
monitor-period	push:millis	The multiplexer progress monitoring period. A watchdog task checks the multiplexer every period. If the	0	1

Name	Type	Description	Min occurs	Max occurs
		multiplexer has not completed at least one operational cycle in this time, a diagnostic warning will be logged to the server log. The default value is 5000 (5 seconds).		
max-event-queue-size	push:positiveInt	The maximum number of entries in the multiplexer event queue. The default value is 128k. Under normal circumstances this value should not be changed from the default.	0	1

hooks

User hooks used in the server.

The following table lists the elements that an element of type `hooks` can contain:

Name	Type	Description	Min occurs	Max occurs
startup-hook	push:string	This is the class name of a class that implements the interface com.pushtechnology.diffusion.api.publisher.ServerStartupHook. If specified, the hook is instantiated and the serverStarting method called when the server is starting.	0	1
shutdown-hook	push:string	This is the class name of a class that implements the interface com.pushtechnology.diffusion.api.publisher.ServerShutdownHook. If specified, the hook is instantiated and the serverStopping method called when the server is stopping.	0	1

security

Server security properties.

The following table lists the elements that an element of type `security` can contain:

Name	Type	Description	Min occurs	Max occurs
authorisation-handler-class	push:string	DEPRECATED: Since 6.5. This element is ignored and should be removed from your configuration.	0	1
authentication-handlers	authentication-handlers		0	1

authentication-handlers

Authentication handlers, in order of decreasing precedence. The authentication handlers are called to authenticate new connections and changes to the principal associated with a session. Authentication handlers are configured in precedence order. Authentication succeeds if a handler returns "allow" and

all higher precedence handlers (earlier in the order) return "abstain". Authentication fails if a handler returns "deny" and all higher precedence handlers return "abstain". If all authentication handlers return "abstain", the request is denied. After the outcome is known, the server might choose not to call the remaining handlers.

The following table lists the elements that an element of type `authentication-handlers` can contain:

Name	Type	Description	Min occurs	Max occurs
authentication-handler	server-authentication-handler	An authentication handler hosted by the server.	0	unbounded
control-authentication-handler	control-authentication-handler	An authentication handler registered by a client.	0	unbounded
system-authentication-handler	system-authentication-handler	An authentication handler that uses the configured system authentication store to validate principals and to define an action for anonymous logins. The XSD does not prevent you from configuring the system authentication multiple times. However, the Diffusion server restricts this and will not start if you define the system authentication handler more than once.	0	unbounded

server-authentication-handler

An authentication handler hosted by the server. The handler is instantiated when the server starts.

The following table lists the attributes that an element of type `server-authentication-handler` can have:

Name	Type	Description	Required
class	push:string	The class attribute specifies the fully qualified name of a handler implementation class that implements the com.pushtechnology.diffusion.client.security.authentication.Authenticator interface. The class must be available on the classpath.	true

system-authentication-handler

The system authentication handler uses the configured system authentication store to validate principals and to define an action for anonymous logins. If the system handler is specified then it will check if a principal is specified in the store and if so will validate its credentials against the store. The store may also specify additional assigned roles to be granted to a principal. If a principal is not specified in the store then the handler will abstain. The store may indicate whether to allow, deny or abstain for anonymous logins.

The following table lists the attributes that an element of type `system-authentication-handler` can have:

Name	Type	Description	Required
hash-scheme	push:string	The hash scheme used to store newly created passwords. More complex algorithms generate password hashes that are more expensive for an attacker to crack by brute force, but require more CPU for each authentication operation. The following schemes are supported: "NONE"; "DESEDE"; "PBKDF-SHA1-N", where N is the number of iterations; "PBKDF-SHA256-N", where N is the number of iterations. The default scheme is PBKDF-SHA256-1000. The configured scheme must be supported by the JVM or the server will not start. All of the above schemes are supported by the standard Java 8 JDK.	false

control-authentication-handler

Client sessions register control authenticators using an identifying name. A <control-authentication-handler> must be configured with a matching handler-name. Configure at most one <control-authentication-handler> for a handler-name.

The following table lists the attributes that an element of type `control-authentication-handler` can have:

Name	Type	Description	Required
handler-name	push:string	The handler name attribute must match the identifying name used by the client session to register a control authenticator.	true

client-queues

Client queue definitions.

The following table lists the elements that an element of type `client-queues` can contain:

Name	Type	Description	Min occurs	Max occurs
default-queue-definition	push:string	The name of the queue definition to use by default. Connectors that do not explicitly specify a queue definition use the one specified here.	1	1
queue-definition	queue-definition	Queue definition.	1	unbounded

queue-definition

This defines the properties of a client queue.

The following table lists the attributes that an element of type `queue-definition` can have:

Name	Type	Description	Required
name	push:string	The queue definition name.	true

The following table lists the elements that an element of type `queue-definition` can contain:

Name	Type	Description	Min occurs	Max occurs
max-depth	push:positiveInt	The maximum depth of the queue. If the number of messages queued for a client exceeds this number, the server disconnects the client. If not set then no limit is applied.	0	1
max-bytes	push:long-bytes	The maximum byte depth of the queue. If the number of bytes queued for a client exceeds this number, the server disconnects the client. If not set then no limit is applied.	0	1
conflates	push:boolean	Specifies whether conflation is enabled by default for queues that use this queue definition. If this value is not specified, conflation is enabled. Conflation can be enabled or disabled for individual sessions at runtime. The behavior when conflation is enabled is determined by the conflation policies of topics.	0	1
upper-threshold	push:percent	This specifies a percentage of the maximum queue size and if this value is reached then any listeners are notified. Notification occurs only once and does not occur again until the queue has returned to the lower threshold. If this value is not specified, no upper limit notification occurs.	0	1
lower-threshold	push:percent	This specifies a percentage of the maximum queue size and indicates the level at which listeners are notified after an upper limit notification has occurred and the queue size has dropped back to the specified lower limit. If this value is not specified, no lower limit notification occurs.	0	1

connection-timeouts

Connection-related timeouts.

The following table lists the elements that an element of type `connection-timeouts` can contain:

Name	Type	Description	Min occurs	Max occurs
write-timeout	push:millis	The write timeout in milliseconds for blocking write operations. Most write operations are non-blocking and are not affected by this timeout. Blocking writes include connection responses to new clients, and HTTP responses to web server requests. If this value is not	0	1

Name	Type	Description	Min occurs	Max occurs
		specified, a default of 3 seconds is used. If this exceeds one hour (3600000ms) a warning will be logged and the time-out will be set to one hour.		
connection-timeout	push:millis	The time in milliseconds allowed for a connection to complete its handshake processing, including the time taken to call any configured authentication handlers and look up location details. If this value is not specified, a default of 5 seconds is used. If this exceeds one hour (3600000ms) a warning will be logged and the time-out will be set to one hour.	0	1

date-formats

Date and time formats.

The following table lists the elements that an element of type `date-formats` can contain:

Name	Type	Description	Min occurs	Max occurs
date	push:string	The format used when displaying dates. Specify the format according to the Java SimpleDateFormat specification. If a format is not specified, a default of "yyyy-MM-dd" is used.	0	1
time	push:string	The format used when displaying times. Specify the format according to the Java SimpleDateFormat specification. If a format is not specified, a default of "HH:mm:ss" is used.	0	1
date-time	push:string	The format used when displaying date and time. Specify the format according to the Java SimpleDateFormat specification. If a format is not specified, a default of "yyyy-MM-dd HH:mm:ss" is used.	0	1
timestamp	push:string	The format used when displaying a timestamp - for example, in a log - to millisecond precision. Specify the format according to the Java SimpleDateFormat specification. If a format is not specified, a default of "yyyy-MM-dd HH:mm:ss.SSS" is used.	0	1

thread-pools

Thread pools.

The following table lists the elements that an element of type `thread-pools` can contain:

Name	Type	Description	Min occurs	Max occurs
inbound	push:string	Name of the inbound thread pool definition.	1	1
background-thread-size	push:int	Number of threads to use for the background thread pool. If a value is not specified, a default of 10 is used.	0	1
thread-pool-definition	thread-pool-definition	Thread pool definition.	1	unbounded

thread-pool-definition

Thread pool definition.

The following table lists the attributes that an element of type `thread-pool-definition` can have:

Name	Type	Description	Required
name	push:string	Name of the thread pool definition.	true

The following table lists the elements that an element of type `thread-pool-definition` can contain:

Name	Type	Description	Min occurs	Max occurs
size	push:positiveInt	The size of the thread pool.	0	1
core-size	push:positiveInt	DEPRECATED: since 6.5. Use size instead, which takes precedence over this setting.	0	1
max-size	push:positiveInt	DEPRECATED: since 6.5. This setting is ignored and should be removed from your configuration.	0	1
queue-size	push:positiveInt	The thread pool queue size. When the max-size is reached, tasks are queued. If the value is 0, the queue is unbounded. If the value is not 0, it must be at least 10.	1	1
keep-alive	push:millis	DEPRECATED: since 6.5. This setting is ignored and should be removed from your configuration.	0	1
rejection-handler-class	push:string	DEPRECATED: since 6.5. This setting is ignored and should be removed from your configuration.	0	1

selector-thread-pools

Selector Thread pools. By default a single pool called "SelectorThreadPool" of size 1 is set up

The following table lists the elements that an element of type `selector-thread-pools` can contain:

Name	Type	Description	Min occurs	Max occurs
default	push:string	Name of the default selector thread pool definition. If not specified then "SelectorThreadPool" is assumed.	0	1
selector-thread-pool-definition	selector-thread-pool-definition	Selector thread pool definition.	1	unbounded

selector-thread-pool-definition

Selector thread pool definition.

The following table lists the attributes that an element of type `selector-thread-pool-definition` can have:

Name	Type	Description	Required
name	push:string	Name of the selector thread pool definition.	true

The following table lists the elements that an element of type `selector-thread-pool-definition` can contain:

Name	Type	Description	Min occurs	Max occurs
size	push:positiveInt	The number of selector threads to have running in the thread pool. The number of selector threads created is the maximum of the value defined here and the number of acceptors defined in the Connectors.xml file. This number is fixed and does not change at runtime.	0	1

whois

WhoIs service details.

The following table lists the elements that an element of type `whois` can contain:

Name	Type	Description	Min occurs	Max occurs
provider	push:string	Name of the WhoIs provider class that must be on the classpath and must implement the API class WhoIsProvider. If a provider is not specified, WhoIsDefaultProvider is used.	0	1
threads	push:int	The number of background threads that process WhoIs resolver requests. If a value is not specified, a default of 0 is used which means that the service is not started.	0	1

Name	Type	Description	Min occurs	Max occurs
host	push:string	The hostname of a WhoIs provider that adheres to the RFC3912 WhoIs protocol. If a hostname is not specified, a default of "whois.ripe.net" is used.	0	1
port	push:port	The port number that the WhoIs provider listens on. If a value is not specified, the normal value of 43 is used.	0	1
whois-cache	whois-cache	Details of the WhoIs service cache that is used to cache WhoIs lookup results. If a value is not specified, the default values are used.	0	1

whois-cache

Details of the WhoIs service cache that is used to cache WhoIs lookup results.

The following table lists the elements that an element of type `whois-cache` can contain:

Name	Type	Description	Min occurs	Max occurs
maximum	push:int	The maximum size of the WhoIs cache. When the cache size exceeds this number it is tidied. A value of 0 means the cache grows indefinitely unless entries are removed because they have exceeded their retention time. If a value is not specified, a default of 1000 is used.	0	1
retention	push:millis	The time for which WhoIs cache entries are retained before being deleted. A value of 0 means entries are retained indefinitely or until the cache reaches its maximum size. If a value is not specified, a default of 0 is used.	0	1
tidy-interval	push:millis	The interval at which the Whois cache tidier task checks if any cache entries have passed their retention time or if the cache has exceeded its maximum size. This is ignored if both maximum and retention are 0. If a value is not specified, a default of 1 hour is used.	0	1

auto-deployment

Auto deployment details. DEPRECATED: Since 6.5. This element is ignored and should be removed from your configuration.

The following table lists the elements that an element of type `auto-deployment` can contain:

Name	Type	Description	Min occurs	Max occurs
directory	push:string	The name of the automatic deployment directory relative to the Diffusion home directory.	1	1
scan-frequency	push:millis	The frequency at which the deployment directory is scanned for new deployments. If a value is not specified, a default of 5 seconds is used.	0	1

geo-ip

GeoIP details.

The following table lists the attributes that an element of type `geo-ip` can have:

Name	Type	Description	Required
enabled	push:boolean	Set to true to enable GeoIP lookup. This needs to be set to true if you are going to use connection or subscription validation policies. If a value is not specified, a default of true is used.	false

The following table lists the elements that an element of type `geo-ip` can contain:

Name	Type	Description	Min occurs	Max occurs
file-name	push:string	The name of the Maxmind GeoCityIP city file. If a value is not specified, the default of "data/GeoLite2-City.mmdb" is used.	0	1

usr-lib

A list of user libraries from which user code is loaded.

The following table lists the elements that an element of type `usr-lib` can contain:

Name	Type	Description	Min occurs	Max occurs
directory	push:string	Directory to load classes from. When the server starts, this folder is traversed, including subdirectories and all jars or zip files added to the class loader.	1	unbounded

fanout

Specifies fan-out connections to establish with primary servers. Typically there is a single connection but it is possible to replicate topics from more than one primary server as long as they do not overlap. All such connections are automatically established when the secondary server starts and will recover as configured.

The following table lists the elements that an element of type `fanout` can contain:

Name	Type	Description	Min occurs	Max occurs
connection	fanout-connection	A fan out connection.	0	unbounded

fanout-connection

Represents a fan-out connection from a secondary server to a primary server.

The following table lists the attributes that an element of type `fanout-connection` can have:

Name	Type	Description	Required
name	push:string	The fanout connection name. If a value is not specified the connection name will be the same as the url value. In a future release this attribute will be mandatory.	false

The following table lists the elements that an element of type `fanout-connection` can contain:

Name	Type	Description	Min occurs	Max occurs
url	push:string	The connection URL which specifies the primary server to connect to.	1	1
principal	push:string	The principal used to connect to the primary server. If not specified, an anonymous connection is assumed.	0	1
password	push:string	The password to use for the connection. If not specified, no credentials are assumed.	0	1
retry-delay	push:millis	This is the time to wait after failing to connect or losing a connection before trying to connect again. The value is specified in milliseconds. If this value is not specified, a default of 1s is used.	0	1
reconnect-timeout	push:millis	This is the total time in milliseconds that will be allowed to reconnect a failed connection to the primary server. For reconnection to work the primary server connector must have been configured to support reconnection. If this is not specified, a value of 60s is assumed. If reconnection is configured and a load balancer is in use then it must be configured for sticky routing.	0	1
recovery-buffer-size	push:int	If the primary server is configured to support reconnection, a session established with a non-zero reconnect-timeout retains a buffer of sent messages. If the session disconnects and reconnects, this buffer is used to re-send messages that the server has	0	1

Name	Type	Description	Min occurs	Max occurs
		not received. The default value is 10,000 messages. If reconnect-timeout is 0 then this value is ignored.		
input-buffer-size	push:bytes	Specifies the size of the input buffer to use for the connection with the primary server. This is used to receive messages from the primary server. Set this to the same size as the output buffer used at the primary server. If not specified, a default of 1024k is used.	0	1
output-buffer-size	push:bytes	The size of the output buffer to use for the connection with the primary server. This is used to send messages to the primary server. Set this to the same size as the input buffer used by the primary server. If not specified, a default of 1024k is used.	0	1
maximum-queue-size	push:int	The maximum number of messages that can be queued to send to the primary server. If this number is exceeded, the connection will be closed. This must be sufficient to cater for messages that may be queued whilst disconnected (awaiting reconnect). The default value is 10,000 messages.	0	1
connection-timeout	push:millis	This specifies the connection timeout value (in milliseconds). If a value is not specified, a default of 2s is used.	0	1
write-timeout	push:millis	This specifies the write timeout value (in milliseconds). If a value is not specified, a default of 2s is used.	0	1
link	fanout-link	Specifies a link to a selection of topics at the primary server that are to be replicated at the secondary server.	1	unbounded

fanout-link

Represents a selection of topics from the primary topic tree to be replicated to the secondary server.

The following table lists the attributes that an element of type `fanout-link` can have:

Name	Type	Description	Required
name	push:string	The fanout link name. If a value is not specified the link name will be the same as the selector value. In a future release this attribute will be mandatory.	false

The following table lists the elements that an element of type `fanout-link` can contain:

Name	Type	Description	Min occurs	Max occurs
selector	push:string	A topic selector specifying the topics to be replicated. This must not overlap (select the same topics as) any other link within this or any other connection configured for the secondary server.	1	1

persistence

Specifies requirements for topic tree persistence. If persistence is enabled, all topics and all updates to those topics will be persisted to append-only log files. Upon restart of the server, persisted topics are restored to their latest persistent state.

The following table lists the attributes that an element of type `persistence` can have:

Name	Type	Description	Required
enabled	push:boolean	Indicates whether the persistence service is enabled. The default is 'true' if the persistence element is present but this attribute is not specified.	false

The following table lists the elements that an element of type `persistence` can contain:

Name	Type	Description	Min occurs	Max occurs
store-directory	push:string	This can optionally be used to specify a directory under which persistence files will be stored. If not specified then files are stored under a directory called 'persistence' within the Diffusion home directory. If specified it should be an absolute directory path under which a directory called 'persistence' will be created. DEPRECATED: since 6.3. Use server persistence-home instead, which takes precedence over this setting.	0	1

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

Configuring connectors

A connector provides a connection point for external applications to connect to the Diffusion server over a TCP connection. Use the `Connectors.xml` configuration file to configure your connectors.

Each connector has a socket server thread which reacts to an incoming connection. The socket information is defined by the connector. Suitable connectors must be defined for inbound connections expected by the Diffusion server.

The following properties are common to all connectors:

Table 43: Connectors properties

Name	A name by which the connector can be identified.
Port	A port number on which to accept requests.
Host	The host to accept requests (only relevant on a multi-homed machine).
Input buffer size	The size of the socket input buffer to use for each connection.
Output buffer size	The size of the socket output buffer to use for each connection.

Socket buffer sizes are very important in achieving the best performance. For more information, see Tuning on page 497.

Client connections

Connectors can accept connections from any type of client. Any number of connectors can be defined to provide different connection points with different properties.

Each client connection has an input buffer to receive messages from the client. The configured input buffer size must be large enough to accommodate the largest message expected from the client. If the maximum message size and the input buffer size are configured as different values, the larger of the two is used as the input buffer size.

The output buffer size is used to assign an output buffer per client multiplexer into which messages are dequeued prior to transmission. This can have an important effect on performance. For more information, see Tuning on page 497.

Start conditions

Connection start conditions prevent the automatic start of a connection until a condition you specify has been met.

These conditions can be useful when you are using load balancers, and you want to ensure that servers do not join the load balancer pool until they are ready.

You can configure start conditions using the `<start-conditions>` element in the `etc/Connectors.xml` configuration file.

The following start conditions are supported:

fanout-ready

The connection does not start until a fan-out link has connected and synchronized.

cluster-configuration-recovered

The connection does not start until the server has joined the cluster and completed restoration of shared configuration.

replicated-topics-recovered

The connection does not start until the server has joined the cluster and completed restoration of replicated topics.

If a connector has no configured start conditions, it will reject attempts to create new sessions until the cluster-wide configuration has been restored.

Here is an example fan-out start condition configuration:

```
<start-conditions>
```

```
  <fanout-ready>
   <connection>primaryServer</connection>
   <link>primaryServerLink</link>
  </fanout-ready>
</start-conditions>
```

Note that the `connection` and `link` values must correspond to the connection and link names in the fan-out configuration in `Server.xml`.

Enabling session reconnection

Specify a reconnection timeout, maximum queue depth, and recovery buffer size by using the `<reconnect>` element in the `etc/Connectors.xml` configuration file.

Reconnection timeout (`keep-alive`)

How long a disconnected client's session remains available on the server before being closed. By default, this is 300 seconds.

Maximum queue depth (`max-depth`)

Optional maximum limit on the number of messages to queue for a disconnected client session. By default, this is the same as the queue depth for a connected client session, which is defined by the queue definitions in `Connectors.xml` and `Server.xml`.

Recovery buffer size (`recovery-buffer-size`)

The maximum number of sent messages to keep in a buffer. These messages can then be recovered on reconnection.

Here is an example connector configuration:

```
<connector>
 ...
 <reconnect>
  <keep-alive>60s</keep-alive>
  <max-depth>1000</max-depth>
  <recovery-buffer-size>64</recovery-buffer-size>
 </reconnect>
 ...
</connector>
```

Using the above example, a client can reconnect to the server through this connector within 60 seconds of becoming disconnected. While the client is disconnected, up to 1000 messages are queued for it. These messages are delivered to the client when it reconnects. A buffer of up to 64 sent messages are retained in the recovery buffer. When a client reconnects, the Diffusion server uses this buffer to re-send any messages that the client has not received.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

Connectors.xml

This file specifies the schema for the connectors properties.

connectors

Connectors

The following table lists the elements that an element of type `connectors` can contain:

Name	Type	Description	Min occurs	Max occurs
connector	connector	Connector definition	0	unbounded

compression-schemes

The following table lists the elements that an element of type `compression-schemes` can contain:

Name	Type	Description	Min occurs	Max occurs
compression-scheme	compression-scheme		0	unbounded

connector

Connector definition

The following table lists the attributes that an element of type `connector` can have:

Name	Type	Description	Required
name	push:string	The connector name	true

The following table lists the elements that an element of type `connector` can contain:

Name	Type	Description	Min occurs	Max occurs
required	push:boolean	This setting specifies if the connector must start successfully when the server is started. If the connector is required (set to true) and the server can not start it during start-up (after waiting for any start conditions} that are configured), the server is stopped. By default, this is set to true.	0	1
host	push:string	The name or the IP address that the connector binds to. This is optional.	0	1
port	push:port	The port on which the connector accepts connections.	1	1
backlog	push:positiveNonZero	The requested maximum queue length for incoming connection requests. It is rarely necessary to tune this value. Operating System settings must be adjusted in tandem. On Linux, the appropriate setting is net.core.somaxconn. If a value is not specified, a default of 1000 is used.	0	1
socket-conditioning	socket-conditioning	Describes the properties associated with TCP socket connections.	1	1
max-queued-bytes	push:long-bytes	The maximum number of bytes that can be queued for all sessions connected through this connector. If the number of	0	1

Name	Type	Description	Min occurs	Max occurs
		bytes queued for a connector exceeds this number, the server may disconnect clients. If not set then no limit is applied.		
web-server	push:string	If this connector is required to serve HTTP requests, this element specifies a web-server entry in WebServer.xml. If a value is not specified, the connector cannot serve HTTP requests. This is also required in order to be able to service websocket connections.	0	1
validation-policy-file	push:string	The location/name of a connection validation policy file to use for this connector. Applies only to type 'all' or 'client'.	0	1
key-store	key-store-definition	Keystore details for any connector that is to support secure (TLS) connections. If this is not specified, TLS connections are not supported.	0	1
queue-definition	push:string	An optional queue definition to use for this connector. This applies only to connectors of type 'all' or 'client'. The definition must exist in Server.xml. If this is not specified, the default queue definition specified in Server.xml is used.	0	1
reconnect	reconnect	Optional reconnection properties which apply only to connectors that accept 'client' connections. If this is not specified, reconnection of client connections is not supported.	0	1
ignore-errors-from	ignore-errors-from	Specifies addresses from which connection errors can be ignored. This is useful for masking errors that might be reported due to the connector port being pinged by some known external entity.	0	1
thread-pool-definition	push:string	Optionally, this can be used to specify a thread pool definition to be used for this connector to create its own inbound thread pool. If specified, the thread pool definition must exist in Server.xml. If a value is not specified, the default inbound thread pool is used.	0	1
selector-thread-pool-definition	push:string	Optionally, this can be used to specify a selector thread pool definition to be used for this connector to deal with NIO operations. If specified, the selector	0	1

Name	Type	Description	Min occurs	Max occurs
		thread pool definition must exist in Server.xml. If a value is not specified, the default selector thread pool is used.		
system-ping-frequency	push:millis	This indicates the interval at which clients are pinged by the server to ensure that they are still connected. If a response is not received from the client before the expiry of another interval period, the client is assumed to be disconnected. If this is not specified or a value of 0 is supplied, clients are not automatically pinged.	0	1
proxy-protocol	proxyProtocol	Indicates the proxy protocol required for connection. Can have the values 'NONE' or 'HA_PROXY'. The default value is 'NONE'. Only connections with the protocol specified are allowed. On publicly accessible connectors, ensure that this value is set to NONE. 'HA_PROXY' refers to the proxy protocol that was first implemented by HAProxy but it is also supported by others including Amazon's Elastic Load Balancer.	0	1
connection-timeout	push:millis	This is the time in milliseconds allowed for a connection to take place and complete its handshake processing. If this value is not specified for a connector, the value set in Server.xml is used.	0	1
compression-schemes	compression-schemes	The compression schemes supported by this connector. The server will use this setting to select an appropriate compression scheme for each session using this connector based on the capabilities declared by the client. The Java, Android, .NET and JavaScript client libraries all support the zlib compression scheme. A JavaScript client must explicitly download the zlib library; it is packaged separately to reduce the download size of the core library. If the compression-schemes element is missing, all compression schemes are supported by this connector. If the compression-schemes element has no compression-scheme child elements, no compression schemes are supported by this connector. That is, compression is disabled.	0	1

Name	Type	Description	Min occurs	Max occurs
start-conditions	start-condition-list	Optional set of conditions that must be satisfied before the connector is automatically started.	0	1
protocols	protocols	Optional list of client protocols supported by this connector. If this element is absent, all protocols are supported. If this element is present but empty, no protocols are supported.	0	1

socket-conditioning

Describes properties associated with TCP socket connections.

The following table lists the elements that an element of type `socket-conditioning` can contain:

Name	Type	Description	Min occurs	Max occurs
input-buffer-size	push:bytes	Specifies the size of the socket input buffer to use for each connection. If a value is not specified, a default of 128k is used. The greater of this value and the max-message-size set in Server.xml is used when setting the socket input buffer size.	0	1
output-buffer-size	push:bytes	This value specifies the size of the output buffer to use for each connection. This must be large enough to accommodate the largest message to be sent. Messages are 'batched' into this buffer and so the larger the buffer, the more messages can be sent in a single write. If a value is not specified, a default of 128k is used.	0	1
keep-alive	push:boolean	This enables or disables TCP keep-alive. If a value is not specified, a default of true is used.	0	1
no-delay	push:boolean	This enables or disables TCP_NODELAY (disable/enable Nagle's algorithm). If a value is not specified, a default of true is used.	0	1
reuse-address	push:boolean	When a TCP connection is closed the connection can remain in a timeout state for a period of time after the connection is closed (typically known as the TIME_WAIT state or 2MSL wait state). For applications using a well-known socket address or port, it might not be possible to bind a socket to the required SocketAddress if there is a connection in the timeout state	0	1

Name	Type	Description	Min occurs	Max occurs
		involving the socket address or port. Enabling this feature allows the socket to be bound even though a previous connection is in a timeout state. If this value is not specified, the feature is enabled.		

reconnect

Reconnect properties.

The following table lists the elements that an element of type `reconnect` can contain:

Name	Type	Description	Min occurs	Max occurs
keep-alive	push:millis	This specifies the reconnection timeout. During this period a disconnected client can reconnect to the same client session. Messages for the client continue to be queued during this period. The default is 5 minutes, meaning reconnection is enabled. Set this value to 0 to disable reconnection.	0	1
max-depth	push:positiveInt	As messages continue to be queued for a client whilst it is disconnected, this enables you to specify a larger maximum queue size that is used during the period that the client is disconnected. When the client reconnects, the maximum reverts back to its previous size (once any backlog had been cleared). If the specified size is not greater than the current maximum size, this has no effect. If this value is not specified, a default of 0 is used which means that no attempt is made to extend the queue size when a client is disconnected.	0	1
recovery-buffer-size	push:positiveInt	If the keep-alive time is not zero, this connector supports reconnection. For each client connected via this connector, the server will retain a buffer of up to recovery-buffer-size sent messages. If a client disconnects and reconnects, the server uses the buffer to re-send messages that the client has not received. The default value is 128 messages. Higher values increase the chance of successful reconnection, but increase the per-client memory footprint.	0	1

key-store-definition

The keystore definition that allows TLS connection to a connector.

The following table lists the attributes that an element of type `key-store-definition` can have:

Name	Type	Description	Required
mandatory	push:boolean	If this is set to true, all connections must use this keystore and TLS connection is mandatory. If a value is not specified, a default of false is used, meaning that the connector accepts either TLS or non-TLS connections.	false

The following table lists the elements that an element of type `key-store-definition` can contain:

Name	Type	Description	Min occurs	Max occurs
file	push:string	The keystore file path.	1	1
password	push:string	The password for the keystore.	1	1

ignore-errors-from

Some external monitors cause the Diffusion server to log errors, as it is not a valid Diffusion connection. Adding the remote IP address to this list ensure that the errors are not logged.

The following table lists the elements that an element of type `ignore-errors-from` can contain:

Name	Type	Description	Min occurs	Max occurs
ip-address	push:string	An IP address or unknown if the remote IP address is being masked.	1	unbounded

start-condition-list

The following table lists the elements that an element of type `start-condition-list` can contain:

Name	Type	Description	Min occurs	Max occurs
fanout-ready	#reference_connectors/section_	A fanout ready condition is satisfied when a fanout link with the given connection and link names exists, has connected to the primary server, and has synchronized all of the topics on the primary server that match its configuration.	0	unbounded
connection	push:string		0	unbounded
link	push:string		0	unbounded
remote-topic-view-ready	#reference_connectors/section_	A remote topic view ready condition is satisfied when a remote topic view with the given name exists, has connected to	0	unbounded

Name	Type	Description	Min occurs	Max occurs
		the remote server, and has evaluated all of the topics on the remote server that match its configuration.		
name	push:string		0	unbounded
cluster-configuration-restored	#reference_connectivity_section_	A cluster configuration restored condition is satisfied when the server is configured as part of a cluster, has joined the cluster, and has completed restoration of shared configuration.	0	1
replicated-topics-restored	#reference_connectivity_section_	A replicated topics restored condition is satisfied when the server is configured as part of a cluster, has joined the cluster, and has completed restoration of replicated topics.	0	1

protocols

The following table lists the elements that an element of type `protocols` can contain:

Name	Type	Description	Min occurs	Max occurs
protocol	protocol		0	unbounded

connectorType

This value must be a push:string.

The following values are allowed:

- all
- client
- policy

proxyProtocol

This value must be a push:string.

The following values are allowed:

- NONE
- HA_PROXY

compression-scheme

This value must be a push:string.

The following values are allowed:

- ZLIB

protocol

This value must be a push:string.

The following values are allowed:

- DIFFUSION_WEBSOCKET
- DIFFUSION_HTTP_LONG_POLL
- MQTT_TCP
- MQTT_WEBSOCKET

Related reference

Configuring connectors on page 436
A connector provides a connection point for external applications to connect to the Diffusion server over a TCP connection. Use the `Connectors.xml` configuration file to configure your connectors.

Configuring security

The `Security.store` file contains the rules that determine the permissions for each role. The `SystemAuthentication.store` stores the principals recognised by the system authentication handler.

> **Note:** As of version 6.5, the old "TopicPermission" API methods used to update these files have been deprecated. The name was misleading because these methods cover all path-based permissions (see Permissions on page 142 for details). Use the equivalent "PathPermission" method instead.

The Security.store and SystemAuthentication.store contain user-editable scripts with a documented syntax. It is safe to edit these files directly when the server is stopped. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

If the files are edited while the server is running, the changes will not be applied until a restart, and could be overwritten by changes made by clients via the API.

Related concepts

Updating the security store on page 367
A client can use the SecurityControl feature to update the security store. The information in the security store is used by the Diffusion server to define the permissions assigned to roles and the roles assigned to anonymous sessions and named sessions.

Role-based authorization on page 138
Diffusion restricts the ability to perform actions to authorized principals. Roles are used to map permissions to principals.

Security.store

The `Security.store` file defines the security roles and the permissions associated with them. It also defines the default set of roles that are assigned to named or anonymous client sessions.

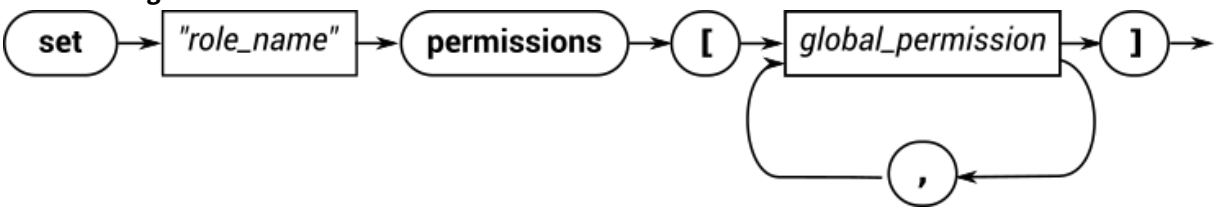
> **Note:** You should stop the server before editing the security store directly. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

The following sections each describe the syntax for a single line of the script file.

> **Note:** The **path** keyword is synonymous with the **topic** keyword used in previous releases of Diffusion. Both keywords are accepted. Prefer **path**.

Assigning global permissions to a role

Railroad diagram



Backus-Naur form

`set "` *role_name* `" permissions ['[' global_permission [ , global_permission ] ']' ]`

Example

```
set "ADMINISTRATOR" permissions [CONTROL_SERVER, VIEW_SERVER,
 VIEW_SECURITY, MODIFY_SECURITY]
set "CLIENT_CONTROL" permissions [VIEW_SESSION, MODIFY_SESSION,
 REGISTER_HANDLER]
```

Assigning default path permissions to a role

Railroad diagram

Backus-Naur form

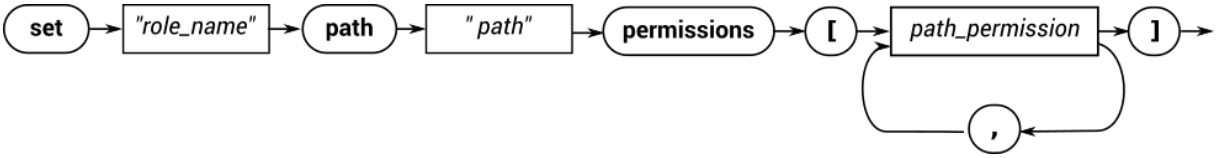
`set "` *role_name* `" default path permissions ['[' path_permission [ , path_permission ] ']' ]`

Example

```
set "CLIENT" default path permissions [READ_TOPIC ,
 SEND_TO_MESSAGE_HANDLER]
```

Assigning path permissions associated with a specific path to a role

Railroad diagram



Backus-Naur form

`set "` *role_name* `" path "` *path* `" permissions ['[' path_permission [ , path_permission ] ']' ]`

Example

```
set "CLIENT" path "foo/bar" permissions [READ_TOPIC,
 SEND_TO_MESSAGE_HANDLER]
set "ADMINISTRATOR" path "foo" permissions [ MODIFY_TOPIC ]
set "CLIENT_CONTROL" path "foo" permissions [ ]
```

Removing all path permissions associated with a specific path to a role

Railroad diagram

Backus-Naur form

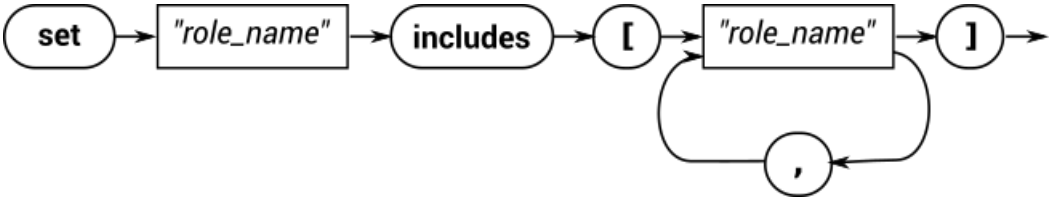
`remove "` *role_name* `" permissions for path "` *path* `"`

Example

```
remove "CLIENT" permissions for path "foo/bar"
```

Including roles within another role

Railroad diagram



Backus-Naur form

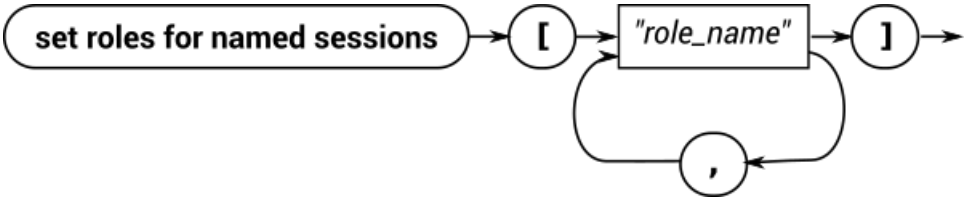
`set "` *role_name* `" includes ['[' "` *role_name* `"[ , "` *role_name* `"]']']`

Example

```
set "ADMINISTRATOR" includes ["CLIENT_CONTROL" , "TOPIC_CONTROL"]
set "CLIENT_CONTROL" includes ["CLIENT"]
```

Assigning roles to a named session

Railroad diagram



Backus-Naur form

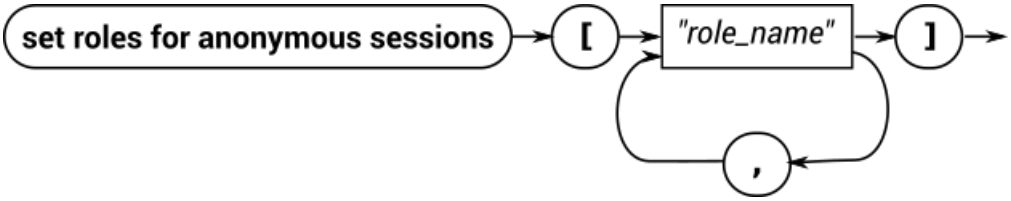
`set roles for named sessions ['[' "` *role_name* `"[ , "` *role_name* `"]']']`

Example

```
set roles for named sessions ["CLIENT"]
```

Assigning roles to an anonymous session

Railroad diagram



Backus-Naur form

```
set roles for anonymous sessions['['"role_name"[ ,"role_name"]']']
```

Example

```
set roles for anonymous sessions ["CLIENT"]
```

SystemAuthentication.store

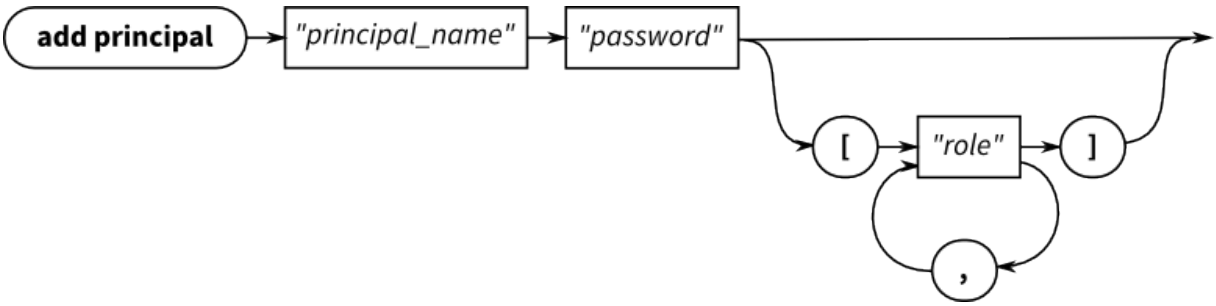
The `SystemAuthentication.store` file defines the roles that are assigned by the system authentication handler to client sessions that have authenticated with a specific security principal. It also defines whether anonymous connections are allowed or denied.

> **Note:** You should stop the server before editing the system authentication store directly. If you are using a cluster, all the servers in the cluster should be stopped before editing. If the server or cluster is running, changes should be made using the management console or the API.

The following sections each describe the syntax for a single line of the file.

Adding a principal

Railroad diagram



Backus-Naur form

```
add principal"principal_name""password"['['"role"[ ,"role"]']']
```

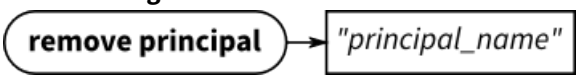
Example

```
add principal "user6" "passw0rd"
add principal "user13" "passw0rd" ["CLIENT", "TOPIC_CONTROL"]
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Removing a principal

Railroad diagram



Backus-Naur form

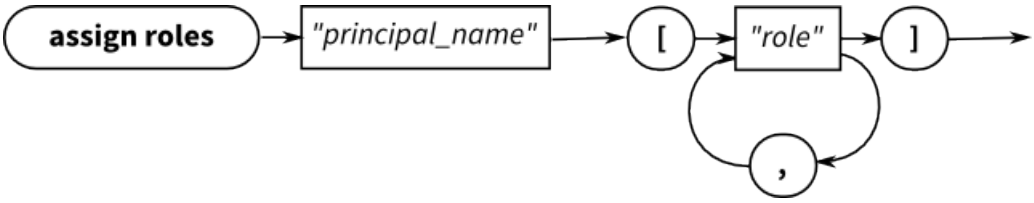
```
remove principal"principal_name"
```

Example

```
remove principal "user25"
```

Assigning roles to a principal

Railroad diagram



Backus-Naur form

`assign roles` " *principal_name* " `'['` " *role* "`[ ,` " *role* "`]']'`

Example

```
assign roles "agent77" ["CLIENT", "CLIENT_CONTROL"]
```

When you use this command to assign roles to a principal, it overwrites any existing roles assigned to that principal. Ensure that all the roles you want the principal to have are listed in the command.

Setting the password for a principal

Railroad diagram

Backus-Naur form

`set password` " *principal_name* " " *password* "

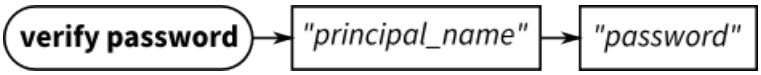
Example

```
set password "user1" "passw0rd"
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Verifying the password for a principal

Railroad diagram



Backus-Naur form

`verify password` " *principal_name* " " *password* "

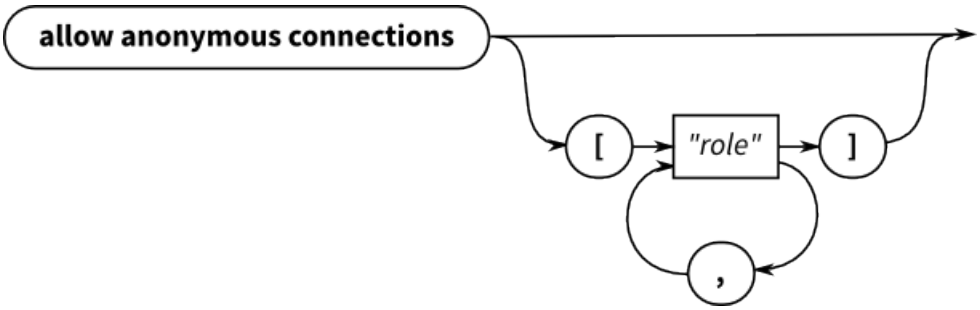
Example

```
verify password "user1" "passw0rd"
```

The password is passed in as plain text, but is stored in the system authentication store as a secure hash.

Allowing anonymous connections

Railroad diagram



Backus-Naur form
`allow anonymous connections['['"`*role*`"[ ,"`*role*`"]']']`

Example

```
allow anonymous connections [ "CLIENT" ]
```

Denying anonymous connections

Railroad diagram

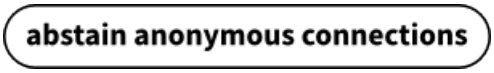
Backus-Naur form
`deny anonymous connections`

Example

```
deny anonymous connections
```

Abstaining from providing a decision about anonymous connections

Railroad diagram



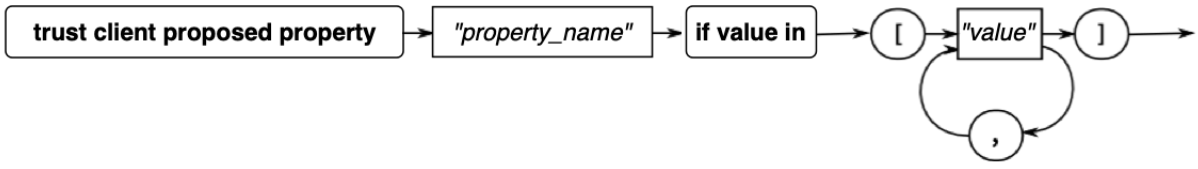
Backus-Naur form
`abstain anonymous connections`

Example

```
abstain anonymous connections
```

Accepting client-proposed session properties with approved values

Railroad diagram



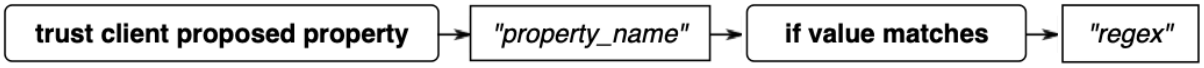
Backus-Naur form
`trust client proposed property"`*property_name*`" '['"`*value*`"[ ,"`*value*`"]']'`

Example

```
trust client proposed property "Foo" if value in ["x", "y", "z"]
```

Accepting client-proposed session properties matching a regex

Railroad diagram



Backus-Naur form

`trust client proposed property "` *property_name* `" if value matches "` *regex* `"`

Example

```
trust client proposed property "Foo" if value matches "^\d{3}-?
\d{2}-?\d{4}$"
```

Use Java-style regular expressions. Evaluation uses `java.util.regex.Pattern`.

Removing a previously-declared trusted client-proposed session property

Railroad diagram

Backus-Naur form

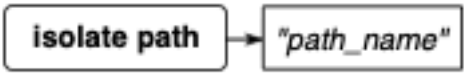
`ignore client proposed property "` *property_name* `"`

Example

```
ignore client proposed property "Foo"
```

Isolating a path from permissions inheritance

Railroad diagram



Backus-Naur form

`isolate path "` *path_name* `"`

Example

```
isolate path "foo/bar/baz"
```

Securing the console

Configuration is required to enable additional security around connections from the Diffusion console.

Allow the console to connect only on a specific connector

We strongly recommend that you only allow the console to connect to Diffusion through a single connector. The port this connector listens on can be blocked from connections from outside of your organization by your load balancer.

You can configure this in the following way:

1. In your `etc/Connectors.xml` configuration file, wherever the line `<web-server>default<web-server>` appears in a connector that receives external connections, replace it with a web server definition that contains only a `client-service` definition. For example:

```
   <web-server name="external">
       <!-- This section enables HTTP-type clients for this Web
Server -->
       <client-service name="client" debug="true">
           <!-- This parameter is used to re-order out-of-order
messages received
                over separate HTTP connections opened by client
browsers. It is rarely
                necessary to set this to more than a few tens of
seconds.
                If you attempt to set this value to more than one
hour, a warning is logged
                and a timeout of one hour is used. -->
           <message-sequence-timeout>4s</message-sequence-timeout>
           <!-- This is used to control access from client web
socket to diffusion.
                This is a REGEX pattern that will match the origin
of the request (.*) matches
                anything so all requests are allowed -->
           <websocket-origin>.*</websocket-origin>
           <!-- This is used to control cross-origin resource
sharing client connection to Diffusion
                This is a REGEX pattern that will match the origin
of the request (.*) matches anything -->
           <cors-origin>.*</cors-origin>
       </client-service>
   </web-server>
```

2. Create a new connector in your `etc/Connectors.xml` configuration file that defines a specific port that you use for internal connections to the console.

   In this connector, set the value of the `web-server` element to default.
3. In your load balancer, prevent outside traffic from having access to the port specified in the new connector.
4. If required, apply additional connection restrictions.
   - You can use a connection validation policy. For more information, see ConnectionValidationPolicy.xml on page 480.
   - You can set these restrictions in your load balancer.

Disable console features in the configuration (as required)

The actions that a user can perform using the console are controlled by roles and permissions. The principal that the user uses to log in to the console must have a role with the permissions required to perform an action in the console.

A principal with the ADMINISTRATOR or OPERATOR role can use all of the functions of the Diffusion console.

To restrict users to using a smaller set of console features, ensure they use a principal with a more restrictive set of roles and permissions. For more information, see Pre-defined roles on page 147.

Configuring logging on the Diffusion server

Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

The Diffusion server uses the JAR file located at `lib/slf4j-binding.jar` as its logging framework. When you first install your Diffusion server, the log4j2 logging framework is used.

Use log4j2

The `log4j-slf4j-impl-`*`version`*`.jar` file controls the log4j2 logging. This file is included in the Diffusion installation in the `lib/thirdparty` directory.

To use log4j2 instead of the legacy Diffusion logging implementation, copy `lib/thirdparty/log4j-slf4j-impl-`*`version`*`.jar` to `lib/slf4j-binding.jar`.

Configure the log4j2 logging framework with the `log4j2.xml` configuration file.

Use the legacy logging framework

To use the legacy Diffusion logging framework, copy `lib/diffusion-slf4j.jar` to `lib/slf4j-binding.jar`.

Configure the legacy logging framework with the `Logs.xml` configuration file.

Use another SLF4J implementation

To use an alternative SLF4J implementation, remove the `lib/slf4j-binding.jar` and add the appropriate classes for the alternative implementation to the Diffusion server classpath.

Note: Alternative implementations of SLF4J are not supported for production use.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Logging back-end on page 543
The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log messages on page 547

The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Logging reference on page 544
Messages logged by the Diffusion server are logged at different levels depending on their severity.

Configuring log4j2

To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

If you are upgrading from a version before 6.4, to use log4j2 instead of the legacy Diffusion logging implementation, copy `lib/thirdparty/log4j-slf4j-impl-*.jar` to `lib/slf4j-binding.jar`. This file controls the log4j2 logging. The Diffusion logging configuration in `etc/Logs.xml` will be ignored.

To revert to the legacy Diffusion logging implementation, copy `lib/diffusion-slf4j.jar` to `lib/slf4j-binding.jar`.

When the Diffusion server is configured to use the log4j2 logging framework, the Diffusion server ignores the configuration in the `Logs.xml` file. Instead, it uses the `log4j2.xml` configuration file.

The `log4j2.xml` configuration file is located in the `etc` directory of your Diffusion installation. For more information about how to use this file to configure log4j2, see the log4j2 documentation: http://logging.apache.org/log4j/2.x/manual/configuration.html

By default, the provided `log4j2.xml` file is configured to output log messages in the format described at Logging reference on page 544. In your configuration file, you can create a property that defines the format to output log messages in:

```
<Property name="pattern">
 %date{DEFAULT_PERIOD}|%level|%thread|%marker|%notEmpty{&lt;
%X{principal}&gt;}|%X{session}|%replace{%msg}{\|}{}|%logger%n%xEx
</Property>
```

You can use this property to specify the format used by your appenders. The property `%marker` indicates the message code. For more information, see Logging reference on page 544.

By default, the provided `log4j2.xml` file is configured to append log output to the console and to a file. This is the same behavior as the legacy logging framework.

```
<Loggers>
        <AsyncRoot level="info" includeLocation="false">
            <AppenderRef ref="console" />
            <AppenderRef ref="file" />
        </AsyncRoot>
    </Loggers>
```

You can configure other appenders to output to the log messages to different destinations. For more information about using appenders, see https://logging.apache.org/log4j/2.x/manual/appenders.html.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework.

This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Logging back-end on page 543
The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log messages on page 547
The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Logging reference on page 544
Messages logged by the Diffusion server are logged at different levels depending on their severity.

Log4j2.xml

Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

```
<Configuration status="warn" name="Diffusion">

    <Properties>
        <Property name="diffusion.log.dir">../logs</Property>

        <!-- The log directory can be be overridden using the
 system property 'diffusion.log.dir'. -->
        <Property name="log.dir">${sd:diffusion.log.dir}</
Property>

        <Property name="pattern">%date{yyyy-MM-dd HH:mm:ss.SSS}|
%level|%thread|%marker|%replace{%msg}{\|}{}|%logger%n%xEx
        </Property>
    </Properties>

    <Appenders>
        <Console name="console">
            <PatternLayout pattern="${pattern}" />
        </Console>

        <RollingRandomAccessFile name="file"
 immediateFlush="false" fileName="${log.dir}/Server.log"
            filePattern="${log.dir}/$${date:yyyy-MM}/Server-%d{MM-
dd-yyyy}-%i.log.gz">

            <PatternLayout pattern="${pattern}" />

            <Policies>
                <OnStartupTriggeringPolicy />
                <TimeBasedTriggeringPolicy />
                <SizeBasedTriggeringPolicy size="250 MB" />
            </Policies>

            <DefaultRolloverStrategy max="20" />
        </RollingRandomAccessFile>
    </Appenders>

    <Loggers>
        <AsyncRoot level="info" includeLocation="false">
            <AppenderRef ref="console" />
            <AppenderRef ref="file" />
```

```
        </AsyncRoot>
    </Loggers>
</Configuration>
```

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Logging back-end on page 543
The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log messages on page 547
The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Logging reference on page 544
Messages logged by the Diffusion server are logged at different levels depending on their severity.

Configuring legacy logging

To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Log messages created by the Diffusion server are filtered by the configuration in `etc/Logs.xml`.

You can configure the following aspects of logging:

- The level of logging to the console
- The level of logging to a file
- The name and location of the file
- Whether the log files rotate based on time or file size or both
- The time interval to use to rotate the files
- The file size to use to rotate the files
- The number of old log files to keep

> **Warning:** Logging can use considerable CPU resources. In a production environment, enable only significant log messages (`INFO` and above). Performance degrades significantly when running at finer logging levels as more messages are produced, each requiring processing.

Logging on the Diffusion server cannot be configured using the Server API. The `LoggingConfig` object is read-only.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Logging back-end on page 543

The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log messages on page 547
The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Logging reference on page 544
Messages logged by the Diffusion server are logged at different levels depending on their severity.

Logs.xml

This file specifies the schema for the log properties used to configure the Diffusion default logging back-end. If you use a different logging back-end, this file is ignored.

logs

Properties defining logging options.

The following table lists the elements that an element of type `logs` can contain:

Name	Type	Description	Min occurs	Max occurs
console-log-level	push:log-level	The log level to start console logging at. Can be ERROR, WARN, INFO, DEBUG, or TRACE. If a value is not specified, a default of INFO is used. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.	0	1
server-log	push:string	The log to use for the server. This must specify the name of a configured log definition. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.	1	1
default-log-directory	push:string	The default log folder for all logs, although this can be overridden for each log. The directory is also used for the daily ConnectionCount statistics file. If not set, the default path is "logs". If the path is relative, it is evaluated relative to the Diffusion installation directory.	1	1

Name	Type	Description	Min occurs	Max occurs
console-monitored-log	push:string	Specifies the log file made available to the Diffusion management console. If not set, the default path is 'Server.log'. If the path is relative, it is evaluated relative to the default log directory.	0	1
async-logging	push:boolean	Indicates whether logging is asynchronous. Asynchronous logging is performed by a separate thread as opposed to being performed in-line by the logging thread. This is normally set to true for performance reasons, but asynchronous logging might cause problems in some OS environments. This element provides the option to turn asynchronous logging off, if so advised. If a value is not specified, a default of true is used. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.	0	1
logging-queue-size	push:positiveInt	The size of the asynchronous logging queue. In normal cases, leave this value at the default value of 128k entries. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.	0	1
thread-name-logging	push:boolean	Indicates whether the thread name is logged with messages. If this is not specified, thread names are logged. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.	0	1
log	log	A log definition. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The	0	unbounded

Name	Type	Description	Min occurs	Max occurs
		Diffusion log implementation will be removed in a future release, together with this configuration setting. Use Log4j 2 instead.		

log

A log definition. DEPRECATED: Since 6.4.0 - This setting is used to configure the Diffusion log implementation, which is deprecated in favor of Log4j 2. The Diffusion log implementation will be removed in a future release, together with these configuration settings. Use Log4j 2 instead.

The following table lists the attributes that an element of type `log` can have:

Name	Type	Description	Required
name		Name of the log definition	true
rotation-period	push:positiveNonZeroInt	A time period that the log exists for before being rotated. This is a positive non-zero integer, with unit specified by rotation-unit. If a rotation-period is specified, the value of file-append must be false.	false
rotation-unit	push:timeunit	A time unit to specify the unit used alongside rotation-period. This can be "day(s)", "hour(s)", "minute(s)".	false

The following table lists the elements that an element of type `log` can contain:

Name	Type	Description	Min occurs	Max occurs
log-directory	push:string	The name of the directory to which this log file is written. If a value is not specified, the default-log-directory is used.	0	1
file-pattern	push:string	This is used to specify the name of the log file. The following values can be used within the pattern. "/" - the local pathname separator. "%t" - the system temporary directory. "%g" - the generation number to distinguish rotated logs. "%h" - the value of the "user.home" system property. "%s" - the system type - for example, 'Diffusion'. "%n" - the system name as defined in Server.xml. "%d" - the date as specified in diffusion.properties (date.format), this is included when using daily rotation. "%%" - translates to a single percent sign "%". If a log file name is not specified, a default of "%s.log" is used.	0	1

Name	Type	Description	Min occurs	Max occurs
level	push:log-level	Specifies the starting log level. This can be ERROR, WARN, INFO, DEBUG, or TRACE. If a value is not specified, a default of INFO is used.	0	1
xml-format	push:boolean	Indicates whether the log file is output in XML format. If a value is not specified, a default of false is used.	0	1
date-format	push:string	Specifies a date format to name a log. Specify the format according to the Java SimpleDateFormat specification. If a format is not specified, a default of "yyyy-MM-dd" is used.	0	1
file-limit	push:bytes	Specifies an approximate maximum amount to write (in bytes) to any one log file. If this is zero, there is no limit. If a value is not specified, a default of 0 is used.	0	1
file-append	push:boolean	Specifies whether log records are appended to existing log files. If a rotation-period is specified, the value of file-append must be false. If a value is not specified, a default of false is used and log files are overwritten.	0	1
file-count	push:positiveNonZeroInt	Specifies the number of log files to use. Must be at least 1. If a value is not specified, a default of 1 is used.	0	1

Logging using another SLF4J implementation

You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

To use an alternative SLF4J implementation, remove the `lib/slf4j-binding.jar` and add the appropriate classes for the alternative implementation to the Diffusion server classpath.

Alternative implementations of SLF4J are not supported for production use.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Logging back-end on page 543
The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log messages on page 547

The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Logging reference on page 544
Messages logged by the Diffusion server are logged at different levels depending on their severity.

Configuring JMX

Use the `Management.xml` configuration file to configure Diffusion to be manageable through JMX. Use the `Publishers.xml` configuration file to configure the JMX adapter to make MBeans available through topics.

Configuring the Diffusion JMX connector server

Connect to JMX through the Diffusion connector server. This connector server is integrated with the Diffusion server and enables you to use role-based access control to define how connecting users can use the MBeans.

About this task

Diffusion binds to the specified ports to listen for connections from JMX clients such as JConsole and Java VisualVM.

Procedure

1. Optional: If you are running Diffusion on a Linux server, check that the host name is not `127.0.1.1`.

   You can do this my running the following command:

   ```
   hostname -i
   ```

   If the output to this command is `127.0.1.1`, add an entry to `/etc/hosts` that defines the host name.
2. Edit the `etc/Management.xml` configuration file to enable and configure the management features:
   a) Set the value of the `enabled` attribute in the `management` element to `true`.

   ```
   <management enabled="true">
   ```

   b) Specify the hostname to allow JMX connections on in the `host` element.

   ```
   <host>localhost</host>
   ```

   The default value is `localhost`. If you set the contents of the `host` element to a value, connections are only allowed to that value. For example, a JMX connection to `localhost` is allowed, but connecting to the same system by IP address is not.

   To allow JMX connections on any applicable hostname or IP address, leave the `host` element blank.

   c) Optional: Specify the ports to use for the JMX service.

   ```
       <!-- The RMI Registry port -->
       <registry-port>1099</registry-port>
       <!-- The JMX service port -->
       <connection-port>1100</connection-port>
   ```

These two ports can be set to the same value, which can simplify firewall configuration.

You can use the default values:

- **1099** The RMI registry port
- **1100** The JMX service port

3. Configure the principals that are allowed to use the JMX service. You can do this in one of the following ways.
   - Update the system authentication store to assign a role with the required permissions to the principal and configure the Diffusion server to call the system authentication handler.

     For more information, see System authentication handler on page 154.
   - Implement a custom authentication handler that assigns a role with the required permissions to the principal and configure the Diffusion server to call your custom authentication handler.

     For more information, see User-written authentication handlers on page 153.
4. **Note:** If you are using a firewall that employs NAT, you might still be unable to connect to Diffusion even when the JMX ports are left open.

   Optional: To make a secure connection or a connection through a firewall, you can use SSH tunnelling:

   a) Establish an SSH connection to the fire-walled Diffusion server.
   b) Tunnel the RMI registry port and JMX service port through SSH.
   c) Use JMX to connect to the local ends of the tunneled ports.

Results

Use the ports you have configured to connect a JMX management console to the Diffusion server.

This connection cannot be made through SSL. However, you can use SSH tunnelling to secure your connection. For more information, see step 4 on page 463.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Using Java VisualVM on page 512
You can manage Diffusion using the JMX system management console Java VisualVM.

Using JConsole on page 514
You can manage Diffusion using the JMX system management console JConsole.

Configuring a remote JMX server connector

Connect to JMX through a remote connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

About this task

Important: We recommend that you use the Diffusion connector server to connect to the JMX service. For more information, see Configuring the Diffusion JMX connector server on page 462.

The JVM that runs Diffusion accepts remote connections from JMX clients such as JConsole and Java VisualVM.

Procedure

1. Configure security for your remote JMX connection.

   For more information, see https://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html.

   The security users and roles defined for the JVM do not integrate with the security provided by the Diffusion server
2. When starting Diffusion, set the properties required for your remote JMX connection.

   For more information, see https://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html.
3. **Note:** If you are using a firewall that employs NAT, you might still be unable to connect to Diffusion even when the JMX ports are left open.

   Optional: To make a secure connection or a connection through a firewall, you can use SSH tunnelling:

   a) Establish an SSH connection to the fire-walled Diffusion server.
   b) Tunnel the RMI registry port and JMX service port through SSH.
   c) Use JMX to connect to the local ends of the tunneled ports.

Results

Use the ports you have configured to connect a JMX management console to the Diffusion server. These connections can be made over SSL.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Using Java VisualVM on page 512
You can manage Diffusion using the JMX system management console Java VisualVM.

Using JConsole on page 514
You can manage Diffusion using the JMX system management console JConsole.

Configuring a local JMX connector server

Connect to JMX through a local connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

About this task

The JVM that runs Diffusion accepts local connections from JMX clients such as JConsole and Java VisualVM.

Procedure

Review the JVM documentation for any actions to take before connecting your JMX client.

For more information, see https://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html.

Results

You can connect a JMX management console running on the same server as Diffusion to the JVM.

Related concepts

JMX on page 511

You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Using Java VisualVM on page 512
You can manage Diffusion using the JMX system management console Java VisualVM.

Using JConsole on page 514
You can manage Diffusion using the JMX system management console JConsole.

Management.xml

This file specifies the schema for the management properties that enable JMX access over an RMI JMXConnectorServer.

management

The management configuration.

The following table lists the attributes that an element of type `management` can have:

Name	Type	Description	Required
enabled	push:boolean	Specifies if an RMI JMXConnectorServer is enabled, making JMX remotely available.	true

The following table lists the elements that an element of type `management` can contain:

Name	Type	Description	Min occurs	Max occurs
host	push:string	The local interface used for the RMI registry and the JMX service. Empty values declare that the RMI registry binds to all local network interfaces.	0	1
registry-port	push:port	The RMI registry port. If a value is not specified, a default of 1099 is used.	0	1
connection-port	push:port	The JMX service port. If a value is not specified, a default of 1100 is used.	0	1
key-store	key-store-definition	Keystore details for any connector that is to support secure (TLS) connections. If this is not specified, TLS connections are not supported.	0	1

key-store-definition

The keystore definition that allows TLS connection to a connector.

The following table lists the elements that an element of type `key-store-definition` can contain:

Name	Type	Description	Min occurs	Max occurs
file	push:string	The keystore file path.	1	1
password	push:string	The password for the keystore.	1	1

Configuring replication

Use the `Replication.xml` configuration file to configure the Diffusion server to replicate sessions and topics.

You can also use the `hazelcast.xml` configuration file to configure your datagrid provider.

Configuring the Diffusion server to use replication

You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

About this task

Ensure that you use the same replication configuration on all of the Diffusion servers in your cluster.

Ensure that each server in the cluster has a unique name, as set in `etc/Server.xml` or the host name if not set.

Configuration items (topic views, metric collectors, and the security/system authentication stores) are replicated if any form of replication is enabled.

Procedure

1. Edit the `Replication.xml` file to configure replication.

```
<replication>
    <provider>HAZELCAST</provider>

 <customConfigurator/>

 <connector>High Volume Connector<connector>

    <sessionReplication enabled="true" />

 <topicReplication enabled="true">
        <topics>
         <excludes>Diffusion</excludes>
        </topics>
    </topicReplication>
</replication>
```

- In the `sessionReplication` element, set `enabled` to `true` to configure the server to replicate sessions between servers in the cluster.
- In the `topicReplication` element, set `enabled` to `true` to configure the server to replicate topics between servers in the cluster.
- If either session or topic replication is enabled, configuration items are replicated. To enable configuration replication without session/topic replication, add a `configurationReplication` element and set its `enabled` property to `true`.
- Inside the `topics` element, use either `includes` or `excludes` elements to define the topics for topic replication and failover of the active update source.

  You can use one or more `includes` elements, or one or more `excludes` elements. You cannot mix `includes` and `excludes`.

  Each element should contain a path identifying a branch of the topic tree.

  If you use `includes` elements, topic replication and failover are applied to the specified branches.

If you use `excludes` elements, topic replication and failover are applied to all topics except those belonging to the specified branches.

Unlike a topic selector, the topic path does not contain any leading or trailing characters. For example, use `<includes>foo/bar</includes>` to select all topics in the branch `foo/bar`.

2. In the `etc/Connectors.xml` file, check there is a connector element with the same name as the connector specified in `Replication.xml`.

   By default, `Connectors.xml` contains a "High Volume Connector" profile which you can use for replication. You should tune the profile based on your particular requirements.
3. Consider adding a `quorum` element inside the `replication` element. This defines a minimum number of servers in a cluster, below which all servers will shut down.

   Using this setting can prevent issues after a network partition separates a cluster, and the two resulting clusters try to rejoin, leading to inconsistent data (a "split-brain" condition). See Server clusters for high availability on page 105 for details.
4. Restart the Diffusion server to load the configuration.
5. Ensure that your clients are configured to reconnect if they lose their connection to the server.

Related reference

Session replication on page 107
You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

Topic replication on page 111
You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Configuring the Hazelcast datagrid on page 467
You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Replication.xml on page 469
This file specifies the schema for the replication properties.

Configuring the Hazelcast datagrid

You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Configuring Hazelcast

By default, the Hazelcast node in your Diffusion server multicasts to all other Hazelcast nodes in the network.

For security, you should consider defining a VLAN or VPC to prevent unwanted Hazelcast multicast connections.

We recommend that, in a production environment, you disable multicast and explicitly define the nodes in your Hazelcast cluster. This configuration is more secure and removes the risk of nodes in your development environment connecting to the production environment and interfering with the production data.

To define which Hazelcast nodes can communicate with each other, use the `hazelcast.xml` configuration file.

We recommend you configure Hazelcast as a mesh (where every node can connect to the others), not a chain (where each node only connects to one other).

The following example shows the structure of the `hazelcast.xml` file:

```
<hazelcast xsi:schemaLocation="http://www.hazelcast.com/schema/config
    http://www.hazelcast.com/schema/config/hazelcast-config-4.0.xsd"
    xmlns="http://www.hazelcast.com/schema/config" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">

    <properties>
        <property name="hazelcast.logging.type">slf4j</property>
        <property name="hazelcast.phone.home.enabled">false</
property>
    </properties>

    <network>
          <join>
            <multicast enabled="false" />
            <tcp-ip enabled="true">
                <member>node1.example.com</member>
                <member>203.0.113.1</member>
                <member>203.0.113.2:5757</member>
                <member>203.0.113.3-7</member>
            </tcp-ip>
        </join>

    </network>
</hazelcast>
```

This example configuration disables the multicast capability and defines the Hazelcast nodes that can be connected to.

The Hazelcast nodes can be defined by hostname, by IP address, or by IP range. The default port used by Hazelcast is 5701. If you want to connect on a different port, you can specify this when you define the node, using the format *host*:*port*.

Ensure that the `hazelcast.xml` file is on the Diffusion server classpath. For example, by putting the file in the *`diffusion_installation`*`/etc` directory. Restart the Diffusion server to load the configuration.

For more information about using the `hazelcast.xml` file to configure Hazelcast, see the Hazelcast™ Reference Manual.

Diagnosing problems with Hazelcast

If you enable logging for Hazelcast, you can use the log files to diagnose problems with Hazelcast.

To enable logging, include the following line in your `hazelcast.xml` file:

```
<property name="hazelcast.logging.type">slf4j</property>
```

Ensure that the `hazelcast.xml` file is on the Diffusion server classpath. For example, by putting the file in the *`diffusion_installation`*`/etc` directory. Restart the Diffusion server to load the configuration.

You can also enable logging by starting the Diffusion server that contains the node with the following parameter `-Dhazelcast.logging.type=slf4j`

You can enable JMX for your Hazelcast nodes and use a JMX tool to examine the MBeans.

To enable JMX for a Hazelcast node, include the following line in your `hazelcast.xml` file:

```
<property name="hazelcast.jmx">true</property>
```

Ensure that the `hazelcast.xml` file is on the Diffusion server classpath. For example, by putting the file in the *`diffusion_installation`*`/etc` directory. Restart the Diffusion server to load the configuration.

You can also enable JMX by starting the Diffusion server that contains the node with the following parameter `-Dhazelcast.jmx=true`

For more information about using Hazelcast, see the Hazelcast™ Reference Manual.

Related tasks

Configuring the Diffusion server to use replication on page 466
You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

Related reference

Session replication on page 107
You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

Topic replication on page 111
You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Replication.xml on page 469
This file specifies the schema for the replication properties.

Replication.xml

This file specifies the schema for the replication properties.

replication

Properties defining replication.

The following table lists the attributes that an element of type `replication` can have:

Name	Type	Description	Required
kubernetes-enabled	push:boolean	Specifies whether Kubernetes replication configuration is enabled for this server. Defaults to 'false'.	false

The following table lists the elements that an element of type `replication` can contain:

Name	Type	Description	Min occurs	Max occurs
provider	push:string	The type of replication provider to use to replicate the data. Currently only Hazelcast is supported.	1	1

Name	Type	Description	Min occurs	Max occurs
connector	push:string	The name of the connector used to configure connections to other servers in the cluster. A connector with the corresponding name should be configured in Connectors.xml. The connector determines the listen host and port, buffer sizes, and the outbound queue size. If this element is not set, the first configured connector will be used instead and a warning will be issued. The connector must be configured as required to start the server (the default behavior) and not have any start conditions.	0	1
external-host	push:string	Optional override of the host name peer servers should use to connect to this server. If not specified, the host name will be derived from the configured connector. The external-host and external-port attributes allow for deployments to environments that use network address translation.	0	1
external-port	push:port	Optional override of the port peer servers should use to connect to this server. If not specified, the port will be derived from the configured connector. The external-host and external-port attributes allow for deployments to environments that use network address translation.	0	1
quorum	push:int	Optional quorum size. Diffusion will only start once the size of the cluster satisfies the quorum. Diffusion will shutdown if the size of the cluster falls below the quorum. The quorum size must be at least two.	0	1
sessionReplication	sessionReplication	The definition for session replication	1	1
topicReplication	topicReplication	The definition for topic replication	1	1
configurationReplication	configurationReplication	The definition for configuration replication. If configuration replication is enabled, configuration items (including topic views, metric collectors, security stores, remote servers) are replicated, even if session and topic replication are disabled. If either session or topic replication is enabled, configuration items are replicated, regardless of this setting.	0	1

sessionReplication

Properties defining session replication.

The following table lists the attributes that an element of type `sessionReplication` can have:

Name	Type	Description	Required
enabled	push:boolean	Specifies whether session replication is enabled for this server.	true

topicReplication

Properties defining topic replication.

The following table lists the attributes that an element of type `topicReplication` can have:

Name	Type	Description	Required
enabled	push:boolean	Specifies whether topic replication is enabled for this server.	true

The following table lists the elements that an element of type `topicReplication` can contain:

Name	Type	Description	Min occurs	Max occurs
topics	topics	The topics that are configured to use replication.	1	1

configurationReplication

Properties defining configuration replication.

The following table lists the attributes that an element of type `configurationReplication` can have:

Name	Type	Description	Required
enabled	push:boolean	Specifies whether configuration replication is enabled for this server. Setting this to true enables configuration items to be replicated regardless of whether session and/or topic replication is enabled. If session or topic replication is enabled then configuration items will be replicated anyway, so this setting would be ignored.	true

topics

Properties defining the topics to replicate. Paths must all be either inclusive or exclusive. It is not possible to include some paths and exclude others.

The following table lists the elements that an element of type `topics` can contain:

Name	Type	Description	Min occurs	Max occurs
includes	push:string	A path that identifies the root of a tree that will be replicated by this server. The path can be of any depth.	0	unbounded

Name	Type	Description	Min occurs	Max occurs
excludes	push:string	A path that identifies the root of a tree that will not be replicated by this server. The path cannot be more than one level deep.	0	unbounded

Related tasks

Configuring the Diffusion server to use replication on page 466
You can configure replication by editing the `etc/Replication.xml` files of your Diffusion servers.

Related reference

Session replication on page 107
You can use session replication to ensure that if a client connection fails over from one server to another the state of the client session is maintained.

Topic replication on page 111
You can use topic replication to ensure that the structure of the topic tree, topic definitions, and topic data are synchronized between servers.

Failover of active update sources on page 112
You can use failover of active update sources to ensure that when a server that is the active update source for a section of the topic tree becomes unavailable, an update source on another server is assigned to be the active update source for that section of the topic tree. Failover of active update sources is enabled for any sections of the topic tree that have topic replication enabled.

Configuring the Hazelcast datagrid on page 467
You can configure how the built-in Hazelcast datagrid replicates data within your solution architecture.

Configuring the Diffusion web server

Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Diffusion can act as a web server by modifying the `Connectors.xml` configuration file to add a `web-server` definition to a connector. If a connector is required to serve HTTP requests, the connector requires a `web-server` definition. A valid `web-server` entry must also exist in the `WebServer.xml` configuration file.

The Diffusion web server is a lightweight web server with very basic features. It hosts the Diffusion landing page, management console, and demos.

The Diffusion web server also provides the endpoint for clients connecting to the Diffusion server using HTTP-based transports.

> **Note:** Do not use the Diffusion web server as the host for your production website. Instead use a third-party web server.

For more information about using Diffusion with third-party web servers, see Web servers on page 618.

Related concepts

Configuring Diffusion web server security on page 473
When configuring your Diffusion web server, consider the security of your solution.

Running the Diffusion server inside of a third-party web application server on page 621

Diffusion can run as a Java servlet inside any Java application server.

Hosting Diffusion web clients in a third-party web server on page 620
Host Diffusion web clients on a third-party web server to enable your customers to access them.

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Related reference

WebServer.xml on page 474
This file specifies the schema for the web server properties.

Configuring Diffusion web server security

When configuring your Diffusion web server, consider the security of your solution.

Digest authentication

Digest authentication can be utilized to negotiate credentials with a user's web browser. It is applied to specific directories on your web site. The protection of one directory automatically applies protection to all lower directories as well.

Use the `realms` element in the `WebServer.xml` configuration file to add new realms to a virtual host and to store the user's name and the passwords.

HTTP deployment

You can deploy DAR files to a Diffusion server through a web service. This web service does not run by default, but can be enabled for your test environment by editing the provided `WebServer.xml` configuration file to include the commented out `deploy-service`.

> **Warning:** Access to the deploy web service is not restricted. Do not enable this web service in your production environment unless you restrict access to the *`diffusion-url`*`/deploy` URL by other means, for example through your firewall setup.

Related concepts

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Web servers on page 130

Consider how to use web servers as part of your Diffusion solution.

Related reference

WebServer.xml on page 474
This file specifies the schema for the web server properties.

WebServer.xml

This file specifies the schema for the web server properties.

web-servers

Definitions of one or more web servers.

The following table lists the elements that an element of type `web-servers` can contain:

Name	Type	Description	Min occurs	Max occurs
web-server	web-server	Web server definition.	0	unbounded

web-server

Web server definition.

The following table lists the attributes that an element of type `web-server` can have:

Name	Type	Description	Required
name	push:string	Name of the web server definition.	true

The following table lists the elements that an element of type `web-server` can contain:

Name	Type	Description	Min occurs	Max occurs
client-service	client-service	Optional client service.	0	1
http-service	http-service	HTTP service.	0	unbounded
file-service	file-service	Optional file service.	0	1

virtual-host

Virtual host definiton.

The following table lists the attributes that an element of type `virtual-host` can have:

Name	Type	Description	Required
name	push:string	Virtual host name.	true
debug	push:boolean	Debug flag. Set to true for debugging. Default is false.	false

The following table lists the elements that an element of type `virtual-host` can contain:

Name	Type	Description	Min occurs	Max occurs
host	push:string	Specifies the host which the virtual host is to serve, for example, download.pushtechnology.com or * for all.	1	1
document-root	push:string	The physical directory for this virtual host. If a relative path is configured, it is resolved relative to the Diffusion home directory.	1	1
home-page	push:string	The default home page. This file is used with directory browsing.	1	1
error-page	push:string	This is used to control the 404 response. The server looks for one of these files in the directory of the request. If the file does not exist, it looks for this file in the virtual directory. If the file is not supplied or the file does not exist, a standard 404 response HTML document is sent.	0	1
static	push:boolean	If this is set to true, after loading the resource once, the file system is not checked again. This improves performance for simple static usage. By default this is false.	0	1
minify	push:boolean	Set to true to minify the html. This happens before the file is compressed. By default this is false.	0	1
cache	cache	The virtual host cache configuration.	1	1
compression-threshold	push:bytes	All HTTP responses over this size are compressed. If not specified, a default value of 512 is used.	0	1
alias-file	push:string	Optionally specifies an alias file. This allows for URL aliasing if required. If a relative path is configured, it is resolved relative to the Diffusion configuration directory.	0	1
realms	realms	Virtual host realms.	0	1

realms

Virtual host realms.

The following table lists the elements that an element of type `realms` can contain:

Name	Type	Description	Min occurs	Max occurs
realm	realm	A virtual host realm.	0	unbounded

realm

A virtual host realm.

The following table lists the attributes that an element of type `realm` can have:

Name	Type	Description	Required
name	push:string	Virtual host realm name.	true
path	push:string	Virtual host realm path.	true

The following table lists the elements that an element of type `realm` can contain:

Name	Type	Description	Min occurs	Max occurs
users	users	Virtual host realm users.	0	1

users

Virtual host realm users.

The following table lists the elements that an element of type `users` can contain:

Name	Type	Description	Min occurs	Max occurs
user	user	Virtual host realm user.	1	unbounded

user

Virtual host realm user.

The following table lists the attributes that an element of type `user` can have:

Name	Type	Description	Required
name	push:string	Virtual host realm user name.	true
password	push:string	Virtual host realm user password.	true

cache

Virtual host cache.

The following table lists the attributes that an element of type `cache` can have:

Name	Type	Description	Required
debug	push:boolean	Set true to debug the cache. If a value is not specified, a default of false is used.	false

The following table lists the elements that an element of type `cache` can contain:

Name	Type	Description	Min occurs	Max occurs
file-size-limit	push:bytes	If the file to be served is over this size, do not cache the entire contents, but map the file instead. If a size is not specified, a default value of 1m is used.	0	1

Name	Type	Description	Min occurs	Max occurs
cache-size-limit	push:bytes	Total size of the cache for this web server definition. If a size is not specified, a default value of 10m is used.	0	1
file-life-time	push:millis	If the file has not been accessed within the time specified, remove the entry from the cache. If a time is not specified, a default value of 1d is used.	0	1

http-service

HTTP service.

The following table lists the attributes that an element of type `http-service` can have:

Name	Type	Description	Required
name	push:string	HTTP service name.	true
debug	push:boolean	Set true to debug the HTTP service. If a value is not specified, a default of false is used.	false

The following table lists the elements that an element of type `http-service` can contain:

Name	Type	Description	Min occurs	Max occurs
class	push:string	The user HTTP service class name. This class must implement the HTTPServiceHandler interface in the web server API.	1	1
url-pattern	push:string	The pattern that the URL must match for this service to be invoked.	1	1
log	push:string	An optional log file can be specified and, if so, HTTP access can be logged. The log definition must exist in Logs.xml.	0	1
max-inbound-request-size	push:bytes	The maximum number of bytes that the HTTP request can have. If this is not specified, a default of the maximum message size is used.	0	1
property	property	HTTP service property.	0	unbounded

property

A property.

The following table lists the attributes that an element of type `property` can have:

Name	Type	Description	Required
name	push:string	Property name.	true
type	push:string	Optional property type.	false

file-service

File service.

The following table lists the attributes that an element of type `file-service` can have:

Name	Type	Description	Required
name	push:string	File service name.	true

The following table lists the elements that an element of type `file-service` can contain:

Name	Type	Description	Min occurs	Max occurs
virtual-host	virtual-host	Virtual host.	1	unbounded
write-timeout	push:millis	Write timeout for serving files. This does not affect HTTP clients. If a value is not specified, a default value of 3s is used.	0	1

client-service

Client service.

The following table lists the attributes that an element of type `client-service` can have:

Name	Type	Description	Required
name	push:string	Client service name.	true
debug	push:boolean	Set true to debug the client service. If a value is not specified, a default of false is used.	false

The following table lists the elements that an element of type `client-service` can contain:

Name	Type	Description	Min occurs	Max occurs
message-sequence-timeout	push:millis	This is used with HTTP clients to indicate how long to wait for a missing message in a sequence of messages before assuming it has been lost and closing the client session. If a value is not specified, a default of 4 seconds is used. If this exceeds one hour (3600000ms) a warning will be logged and the time-out will be set to one hour.	0	1
websocket-origin	push:string	The server will reject a WebSocket connection upgrade request that has an Origin header if the header does not match this regular expression. The default value is ".*", which allows all WebSocket upgrade requests.	0	1
cors-origin	push:string	This is used to control access from client web (XHR) to Diffusion. This element will enable Cross Origin Resource Sharing (CORS). This is a regular expression pattern that matches the origin of	0	1

Name	Type	Description	Min occurs	Max occurs
		the request. A value of ".*" matches anything, so all requests are allowed. If a value is not specified, the service cannot handle CORS requests.		
max-inbound-request-size	push:bytes	The maximum number of bytes that the HTTP request can have. If a value is not specified, a default of the maximum message size is used.	0	1
disable-cookies	push:boolean	Set true to disable session cookie from being in the "Set-Cookie" header. If a value is not specified, cookies are enabled.	0	1

property-value

This value must be a push:string.

Related concepts

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Configuring Diffusion web server security on page 473
When configuring your Diffusion web server, consider the security of your solution.

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Aliases.xml

This file specifies the schema for the aliases properties used in a web server.

aliases

List of aliases

The following table lists the elements that an element of type `aliases` can contain:

Name	Type	Description	Min occurs	Max occurs
alias	alias	An alias definition	0	unbounded

alias

An alias definition

The following table lists the attributes that an element of type `alias` can have:

Name	Type	Description	Required
name	push:string	A name for the alias.	true

The following table lists the elements that an element of type `alias` can contain:

Name	Type	Description	Min occurs	Max occurs
source	push:string	The source URL, which can be expressed as a regular expression.	1	1
destination	push:string	The destination path.	1	1

ConnectionValidationPolicy.xml

This file specifies the schema for the connection validation policy.

connection-validation-policies

Connection validation policies

The following table lists the elements that an element of type `connection-validation-policies` can contain:

Name	Type	Description	Min occurs	Max occurs
policy	policy	A connection validation policy.	0	unbounded

policy

A connection validation policy.

The following table lists the attributes that an element of type `policy` can have:

Name	Type	Description	Required
name	push:string	Each policy must be supplied with a unique name for easy reference.	true
type	push:string	The policy type should be either "blacklist" or "whitelist". A blacklist indicates that if any of the policy rules in this policy match the incoming connection, that connection is to be rejected. A whitelist requires that at least one policy rule matches for the connection to be accepted.	true

The following table lists the elements that an element of type `policy` can contain:

Name	Type	Description	Min occurs	Max occurs
addresses	addresses	Connection validation policy addresses. These are addresses that are blacklisted/whitelisted.	0	1
locale	locale	Connection validation policy locale. This is locale details that are blacklisted/ whitelisted.	0	unbounded

addresses

The following table lists the elements that an element of type `addresses` can contain:

Name	Type	Description	Min occurs	Max occurs
address	push:string	An IP address (or regular expression) of a connecting client.	0	unbounded
hostname	push:string	The hostname (or regular expression) of a connecting client.	0	unbounded
resolved-name	push:string	The resolved hostname (or regular expression) of a connecting client, as returned by the WhoIs service.	0	unbounded

locale

The following table lists the elements that an element of type `locale` can contain:

Name	Type	Description	Min occurs	Max occurs
country	push:string	The ISO country code of the connecting client, as returned by the WhoIs service.	0	1
language	push:string	The ISO language code of the connecting client, as returned by the WhoIs service.	0	1

Env.xml

This file specifies the schema for the environment properties.

env

The following table lists the elements that an element of type `env` can contain:

Name	Type	Description	Min occurs	Max occurs
property	property	Environment variable value	0	unbounded

property

The following table lists the attributes that an element of type `property` can have:

Name	Type	Description	Required
name	xsd:token	Name of the environment variable.	true

propertyValue

This value must be a xsd:token.

Mime.xml

This file specifies the schema for the mime properties.

mimes

The following table lists the elements that an element of type `mimes` can contain:

Name	Type	Description	Min occurs	Max occurs
mime	mime	Mime.	0	unbounded

mime

The following table lists the attributes that an element of type `mime` can have:

Name	Type	Description	Required
type	push:string	Mime type.	true
extension	push:string	Mime extension.	true

mimeValue

This value must be a xsd:string.

Statistics.xml

This file specifies the schema for the properties defining statistics collection.

statistics

Properties defining statistics collection.

The following table lists the elements that an element of type `statistics` can contain:

Name	Type	Description	Min occurs	Max occurs
client-statistics	client-statistics	Control over session statistics. Summary reports will regularly be written to the server log. The log message gives a	0	1

Name	Type	Description	Min occurs	Max occurs
		count of all of the different client types. Each counter is reset according to the configured frequency.		
topic-statistics	topic-statistics	Control over topic statistics. DEPRECATED: since 6.3. Per-topic statistics have been replaced by topic metric collectors. This element will be removed in a future version of the product.	0	1
reporters	reporters	DEPRECATED: since 6.3. Statistics reporters have been removed. This element will be removed in a future version of the product.	0	1

client-statistics

The following table lists the elements that an element of type `client-statistics` can contain:

Name	Type	Description	Min occurs	Max occurs
output-frequency	push:millis	Specifies the output frequency of the log. There is one entry per specified interval. If this is not specified, a default of 1h is used.	0	1
reset-frequency	push:millis	Specifies when the counters are reset. The reset interval must be a multiple of the output frequency. Zero specifies that the counters are never reset. If this is not specified, a default of 1h is used.	0	1
monitor-instances	push:boolean	DEPRECATED: since 6.3. Per-client statistics have been replaced by session metric collectors. This setting is no longer used and will be removed in a future version of the product.	0	1

topic-statistics

The following table lists the elements that an element of type `topic-statistics` can contain:

Name	Type	Description	Min occurs	Max occurs
monitor-instances	push:boolean	Specifies if individual topic statistics are enabled.	0	1

reporters

Cross domain

The `etc` directory contains an additional `crossdomain.xml` XML file. The format of this XML file is not defined by Push Technology.

The following XML file is included in the `etc` directory. For more information, see Cross domain policies on page 625.

`crossdomain.xml`

Use this file to grant a web client permission to handle data across multiple domains.

Starting the Diffusion server

After you have installed and configured your Diffusion server, you can start it using one of a number of methods.

Use the provided Diffusion start scripts

Your Diffusion installation includes The `diffusion.sh` or `diffusion.bat` command (issued in the `bin` directory) starts Diffusion. An optional properties directory can be specified as a parameter to be used instead of the default `../etc` directory.

> **Important:** Do not run your Diffusion server as root on Linux or UNIX. To run the Diffusion server on a port number of 1024 or lower, use another means. For some examples of ways of doing this, see http://www.debian-administration.org/articles/386.

Use a script in `init.d`

On Linux, Diffusion can be started using a script in your `/etc/init.d` folder that starts your Diffusion server when the host server starts.

If you installed your Diffusion server using RPM, this script already exists in your `/etc/init.d` folder.

If you installed your Diffusion using another method, you can use the sample script files in the `tools/init.d` directory of your Diffusion. Edit the sample script file to include the location of your installation and make any other changes that are required. Copy the edited script file to `/etc/init.d`. Ensure that the file is executable.

When your host server starts, it starts your Diffusion server.

Use Docker

Diffusion is provided as a Docker image on DockerHub. When you use Docker to run this image, the Diffusion server inside the image is started.

For more information, see Installing the Diffusion server using Docker on page 405.

Run embedded in a Java process

You can run the Diffusion server from within a Java process by including the `diffusion.jar` on the classpath of the Java process.

For more information, see Running from within a Java application on page 485.

Running from within a Java application

To run Diffusion from within a Java application instantiate, configure and start a `DiffusionServer` object.

Creating a server

`DiffusionServer` is available in the `com.pushtechnology.diffusion.api.server`. You can instantiate it with one of the following constructors:

Default configuration

```
DiffusionServer server = new DiffusionServer();
```

This instantiates the server with default configuration options. The default configuration is read from the XML configuration files in the `etc` directory of your Diffusion installation. Required aspects of the server must be configured before it is started. These can be configured programmatically. For more information, see Programmatic configuration on page 417.

Bootstrap properties

```
DiffusionServer server = new
 DiffusionServer(bootstrapProperties);
```

This specifies a set of properties inside a `Properties` object. The following properties are supported:

diffusion.home	The base installation directory	Calculated from the location of the `diffusion.jar` file.
diffusion.config.dir	The configuration directory, where the XML configuration files are located.	*`diffusion.home`*`/etc`
diffusion.license.file	The license file	*`diffusion.config.dir`*`/license.lic`
diffusion.keystore.file	The keystore file required to decrypt the license	*`diffusion.config.dir`*`/licence.keystore`

These properties can also be set as system properties.

Whichever approach to instantiation that you use, a full set of XML configuration files can be present in the configuration directory and tuned as required or just a partial set of the files can be present and all missing configuration supplied programmatically.

Configuring the Diffusion server

Once the server object has been instantiated some properties can be configured. The root configuration object can be obtained from the server object as follows:

```
ServerConfig config = server.getConfig();
```

Alternatively the root can be obtained using `ConfigManager.getServerConfig()`.

The configuration is populated from the XML configuration files in the installation or in the configuration directory specified by *diffusion.config.dir*. You can further customize the configuration to fit your requirements. For an example, see the following sample code:

```
DiffusionServer server = new DiffusionServer();

ServerConfig config = server.getConfig();


// Connector ConnectorConfig connector = config.addConnector("Client
 Connector");
// Configure connector as required....

// Thread Pools
ThreadsConfig threads = config.getThreads();
ThreadPoolConfig inbound = threads.addPool("Inbound");
inbound.setCoreSize(3);
inbound.setMaximumSize(10);
inbound.setQueueSize(2000);
threads.setInboundPool(inbound.getName());
threads.setBackgroundPoolSize(2);

// Queues QueuesConfig queues = config.getQueues();
QueueConfig queue = queues.addQueue("DefaultQueue");
queue.setMaximumDepth(10000);
queues.setDefaultQueue("DefaultQueue");

// Multiplexer MultiplexerConfig multiplexer =
 config.addMultiplexer("Multiplexer");
multiplexer.setSize(4);
```

Note: The logging configuration cannot be changed using the `ServerConfig` object. By the time the configuration objects are available to your Java application, the logging properties are locked. The `LoggingConfig` object is read-only.

Monitoring the Diffusion server lifecycle

The `DiffusionServer` class provides methods to add and remove a lifecycle listener on the server instance.

```
addLifecycleListener(LifecycleListener stateCallback);
```

```
removeLifecycleListener(LifecycleListener stateCallback);
```

The lifecycle listener is a callback that is called whenever the server state changes. The Diffusion server can have the following states:

INITIAL

An instance of the `DiffusionServer` exists, but has not been started.

STARTING

The server is starting.

STARTED

The server has started.

STOPPING

The server is stopping.

STOPPED

The server has stopped.

Starting the server

After the server configuration has been completed, the server can be started using `server.start()`.

Connectors start to listen on the configured ports.

Stopping the server

The server can be stopped using `server.stop()` at which point the server is no longer available.

Run requirements

A simple way to use Diffusion as a library within your application is to install Diffusion and include the path to `diffusion.jar` in your CLASSPATH.

If you want to repackage the Diffusion code more extensively, be aware of the following concerns:

- `diffusion.jar` depends on other library files in the installation's `lib` directory and are referenced in the jar's manifest `Class-Path` entry. You must also make the code in these libraries available.
- You still require a Diffusion installation. The installation provides the configuration, licence, and log directories. The installation directory is calculated from the location of `diffusion.jar`. If `diffusion.jar` is not loaded from a URL classloader, or has been moved from the product installation, use the bootstrap properties constructor and set the *diffusion.home* system property to the installation directory.

Limitations

Currently only one Diffusion server can be instantiated in a Java VM and it can be started only once.

Network security

This section describes how to deploy network security, which can be used in conjunction with data security.

Secure clients

Diffusion clients can connect to your solution using TLS or SSL.

> **Note:** The secure connection can terminate at your load balancer or at your Diffusion server. Terminating at the load balancer can reduce the CPU load on your Diffusion servers; the supported protocols will depend on your load balancer.

If you are terminating the secure connection at the server, the available TLS/SSL protocols depend on the implementations provided by the Java installation you are using to run the Diffusion server. For a given connection, the client and server will negotiate the protocol and cipher suite to use.

Java distributions track security vulnerabilities and enable protocols and cipher suites accordingly. For example, SSL protocols have known vulnerabilities and are typically disabled.

Use the interfaces provided by your Java installation to override the defaults, and enable or disable particular protocols or cipher suites for the server or the Java client. For the Oracle JDK, see the Oracle JDK cryptographic configuration documentation.

The diffusion.tls.protocols and https.cipherSuites system properties used to configure secure connections in previous Diffusion versions are now ignored.

Diffusion has been tested with a range of protocols, including TLSv1.3. For TLSv1.3 support, we recommend running the server using a Java 11 OpenJDK-based distribution. Azul Zing can be used if TLSv1.3 support is required for Java 8 clients.

See System requirements for the Diffusion server on page 27 for more information about supported Java distributions.

Session tokens

Diffusion clients use session tokens to authenticate when reconnecting to their existing session. To protect the credentials supplied in the original connection request and the returned session token, ensure that the client uses a secure transport to communicate with the Diffusion server. For example, WSS.

Session tokens are generated by using `java.security.SecureRandom` with the default algorithm supplied by the Java environment used to run the Diffusion server. Each token is a 24-character string encoded in base-64, representing 18 bytes (144 bits) of random data.

A new session token is generated when a client connects to the Diffusion server, is authenticated, and creates a session. The server returns the session token to the client in the connection response. The client library keeps the session token in memory. If the client connection is lost, the client attempts to reconnect and supplies the session token. The server is configured with a reconnection timeout. If the Diffusion server detects the loss of the client connection and the client fails to reconnect to the Diffusion server before the reconnection timeout has elapsed, the Diffusion server closes the session and the session token is no longer valid. If the client reconnects before the reconnection timeout has elapsed, the Diffusion server accepts the new connection using the session token is used as proof of authentication.

Web server configuration

The web server can be configured in your test environment to allow you to deploy and undeploy DAR files by using a web service. By default this capability is not enabled.

For security, if you choose to enable this web service in your production environment, you must restrict access to the *`diffusion-url`*`/deploy` URL by other means. For example, by setting up restrictions in your firewall.

To configure the web server, use the `WebServer.xml` file. For more information, see WebServer.xml. An example of this file is provided in the `/etc` directory of the Diffusion installation. The XSD is provided in the `/xsd` directory of the Diffusion installation.

Connector configuration

If secure connections are required, Diffusion connectors must be configured to support HTTPS, WSS, or a combination of these transports. Any connector can accept secure connections. A connector does not have to be dedicated to only secure connections. To enable secure connections a keystore entry is required in the connector configuration. This informs the connector that it is enabled for secure connections. If HTTPS is required, a keystore section and a web-server entry are also required, even for secure Diffusion clients.

To configure the connectors, use the `Connectors.xml` file. For more information, see Connectors.xml . An example of this file is provided in the `/etc` directory of the Diffusion installation. The XSD is provided in the `/xsd` directory of the Diffusion installation.

Keystores

The default Diffusion installation includes a sample keystore containing a self-signed certificate. This is suitable for development. The certificate will not be trusted by browsers and other clients without additional configuration. If you use TLS in production, you must create a new keystore, using a certificate obtained from a certificate authority.

The following steps use the Java Keytool to create a keystore. The steps can vary depending on your certificate authority. For more information, refer to your certificate authority's documentation.

1. Generate a key and place it in your keystore.

```
keytool -genkeypair -alias my_alias -keyalg RSA -
keystore keystore_name -keysize
```

2. Generate a CSR file.

```
keytool -certreq -keyalg RSA -alias my_alias -file certreq.csr -
keystore keystore_name
```

3. Send the CSR file to your certificate authority.
4. Receive the signed certificate from your certificate authority.
5. Install any intermediate certificates that you require.

```
keytool -import -trustcacerts -alias intermediate_alias -
keystore keystore_name -file intermediate_certificate_file.crt
```

6. Install your own certificate. Use the same alias as when you generated the key and the signing request.

```
keytool -import -trustcacerts -alias my_alias -
keystore keystore_name -file certificate_file.crt
```

Provided keystores

The `etc` directory of the Diffusion directory contains the following keystores:

sample.keystore

This keystore is an example keystore that contains a self-signed certificate. In production, we recommend you create your own keystore that contains a certificate signed by a certificate authority.

licence.keystore

This keystore contains the public key used for the Diffusion license file. Do not edit or delete this keystore. Diffusion requires this keystore to verify your Diffusion license.

Related information

https://www.sslsupportdesk.com/java-keytool-commands/

Going to production

When going to production with Diffusion review this information for recommendations on preparing for a successful production deployment.

The advice in this section is not an exhaustive list of steps to take when getting ready to take your Diffusion solution into production. You might have additional requirements based on your solution. Push Technology provides Professional Services that can work with you to advise on a pre-production testing strategy specific to your requirements. Email consulting@pushtechnology.com for more information.

Pre-production testing

The most important part of taking your solution into production is to ensure that you fully test it under as wide a range of expected conditions as possible.

Setting up your test environment

Ensure that the environment you set up to test your solution is as close as possible to the production environment you intend to deploy.

About this task

There are many benefits to creating a test environment that is the same as your production environment:

- It enables you to do regression testing when you change the version or configuration of any of the components in your solution.
- It enables you to test your solution's performance under different levels of stress and load.
- It provides a controlled environment where you can reproduce any issues that you encounter in your production system.
- It provides an environment where you can capture runtime data that cannot easily be captured in production without affecting the behavior or performance of the production system.

To create a test environment that closely reflects your production environment, consider taking the following steps:

Procedure

1. Use the same hardware as you intend to deploy your production solution on.

   Consider the following aspects:

   - Hardware specifications
   - Operating system version and patch version

   For more information, see Requirements.
2. Use the same network specifications as you intend to use in your production solution.

   Consider the following aspects:

   - Speed
   - Connection reliability

   > **Note:** This is especially important when testing mobile applications. The behavior of your mobile client can be different depending on whether the client connects through WiFi or 3G, for example.

3. Use the same JDK version as you intend to use to run your Diffusion server.

   Consider the following aspects:

   - The JVM version
   - Any tuning parameters you intend to use

   For more information, see Requirements.
4. Include third-party components that will be used in your production solution.

   Consider your use of the following components:

   - Load balancers

     For more information, see Load balancers on page 129 and Common issues when using a load balancer on page 630.
   - Web servers

     For more information, see Web servers on page 130 and Web servers on page 618.
   - JMS servers

     For more information, see JMS on page 133 and JMS adapter on page 639.

   There are special considerations for using these third-party components with Diffusion. Ensure that you have fully reviewed the linked documentation for any of the components you are using.
5. Ensure your Diffusion servers use the same version and configuration as in your production solution.

   Consider the following aspects:

   - Diffusion server version
   - Diffusion server configuration
   - Diffusion server security configuration
6. Include all the components you have developed for use in your production system.

   Consider your use of the following components:

   - Clients

     Both those within your organization and those used by your customers.
   - Server-side components

Understanding production usage conditions

Consider the flow of data and the actions of clients in your Diffusion server. Pre-production testing that closely models the usage you expect to see in production is most useful in understanding how your solution will respond in production.

The following sections contain some of the questions to consider when deciding how to test your solution before going to production. For each of these questions consider both average use values and edge case values.

Client connections

- How many clients do you expect to attempt to connect simultaneously?
- How many clients do you expect to be connected concurrently?
- Do you have session replication enabled and, if so, in a failover situation do you expect all of your concurrently connected clients to attempt reconnect at the same time?
- How long is a client connection expected to last?
- What is the expected geographic distribution of client connections?
- How does your load balancer decide how to distribute incoming client connections?
- How are your expected client connections distributed by platform or API?

- How are your incoming client connections authenticated?

Topics

- At what frequency do you expect to create topics?
- How many topics do you expect to create at the same time?
- At what frequency do you expect to delete topics?
- How many topics do you expect to delete at the same time?
- How many topics do you expect your clients to be subscribed to?
- How many topics do you expect your clients to subscribe to in a single action?

Topic updating

- How many topics is a given client expected to update?
- How frequently do you expect topics to be updated?
- How many topics do you expect to send updates to at the same time?
- How many topics do you expect a given client to send updates to at the same time?
- How big do you expect the data in your topic updates to be?

Other client actions

- How many client authentication requests is a given client expected to handle?
- How many messages sent to a message path is a given client expected to handle?
- How many messages sent directly to the client is a given client expected to receive?
- How many messages is a given client expected to send to a message path?
- How many messages is a given client expected to send directly to another client?
- How often do you expect clients to manage other clients?
- How many clients do you expect a given client to manage?

How to create production usage conditions in your test environment

You can create production usage conditions in your test environment by either recording live production usage and playing it back or by simulating production usage.

Recording production conditions

Recording production conditions involves inserting components within your production system to track specific actions. For example:

- A recording tool upstream of the Diffusion server that records the data stream being fed in to Diffusion topics
- A recording tool at your load balancer to record incoming connections, where they come from, when they connect, and how long they remain connected.

After this data has been captured in production test tools use the data to replay or simulate identical conditions in your test environment.

The record and playback approach has the following advantages:

- The data and client actions have occurred in production and reflect realistic production conditions.

The record and playback approach has the following disadvantages:

- You cannot use the recorded data to test conditions that have not occurred in your production environment, but that you might expect to occur.
- You must have an existing production environment to capture data from.
- You must develop the tools to capture and store production conditions. Introducing these components to your production system might effect its behavior.

- You must develop the tools to replay production conditions in your test environment.

Simulating production conditions

Simulating production conditions involves developing tools or test harnesses that generate data or client behavior in your test environment.

The simulation approach has the following advantages:

- You can test a wider range of conditions than those that have occurred in production.

The simulation approach has the following disadvantages:

- There is a risk that the simulation might not create realistic production conditions.
- You must develop the tools to simulate the conditions you want to test.

Using live production data

You can create production conditions in your data by using the same data stream as the production environment uses to feed into your test environment.

The live production data approach has the following advantages:

- The data stream being fed in to Diffusion is real.
- You do not need to create tools to record and playback or to simulate production data.

The live production data approach has the following disadvantages:

- Depending on the type of client information in the production data and your data protection policies and legal requirements, you might not be permitted to use live production data in a test environment.
- Depending on the type of client information in the production data and your data protection policies and legal requirements, you might not be permitted to send certain diagnostics to Push Technology when requesting support.
- You must ensure that nothing in your test environment can affect either the production data or the production environment.
- You must have an existing production environment to use data from.
- If you want to test specific data conditions, you are restricted to doing so at the times when these conditions occur.

Using live production traffic

You can simulate production conditions in your client traffic by duplicating client requests coming into your live production environment in your test environment and by suppressing the responses made by the test server from reaching the production client.

The live production traffic approach has the following advantages:

- The client requests to Diffusion are real.
- You do not need to create tools to record and playback or to simulate production traffic.

The live production traffic approach has the following disadvantages:

- Depending on the type of client information in the traffic and your data protection policies and legal requirements, you might not be permitted to use live production traffic in a test environment.
- Depending on the type of client information in the production traffic and your data protection policies and legal requirements, you might not be permitted to send certain diagnostics to Push Technology when requesting support.
- You must have an existing production environment to use the traffic from.
- You must ensure that responses from your test environment do not reach the client.

- Because the responses from the test environment do not reach the production clients, who instead receive responses from the production environment, the behavior does not accurately reflect server-client interactions.
- If you want to test specific traffic conditions, you are restricted to doing so at the times when these conditions occur.

These techniques can be used separately or together to give the fullest range of test conditions.

Both involve the development of custom tooling to create the required conditions. Push Technology provides Professional Services that can work with you to create these tools. Email consulting@pushtechnology.com for more information.

Types of testing

Consider performing the following types of testing before taking your solution into production.

Component testing

Before setting up your test environment, test the clients and other components that you have developed. For each component, consider doing the following testing:

- Unit testing with a high level of code coverage
- End-to-end testing
- Performance testing
- Stress testing
- Usability and accessibility testing, if the component is customer-facing.

For more information, see Testing on page 395

Smoke testing

On first setting up your test environment, smoke test your solution to ensure that all basic expected function works before proceeding to more in-depth testing.

For more information, see Smoke Testing on Wikipedia

Regression testing

If you have an existing Diffusion solution in production and are updating one or more components or their configuration, perform regression testing in your test environment to ensure that the behavior of your solution has not changed in unintended ways before updating your production environment.

For more information, see Regression Testing on Wikipedia

Load testing

Ensure that you test your Diffusion solution at peak expected load to discover how your solution handles these conditions. This load can be client connections, topics, topic updates, and combinations of load types.

For more information, see Load Testing on Wikipedia

Soak testing

Most Diffusion solutions are expected to run continuously under varying load. Ensure that you test how your solution behaves when it is left in operation for a long time, for example, 24 hours. Long test runs can uncover potential resource leaks, long garbage collections, or previously unforeseen timeouts.

For more information, see Soak Testing on Wikipedia

Failover and recovery testing

If your solution has failover or replication configured on your Diffusion servers, test that these work as you expect when one of the Diffusion becomes unavailable. For resiliency of your whole solution, other components – for example, load balancers – can be configured to failover or provide redundancy. Ensure that these measures work as you expect.

For more information, see Configuring the Diffusion server to use replication on page 466 and Using load balancers for resilience on page 629

Penetration testing

Diffusion provides mechanisms to secure which ports clients can connect to your Diffusion server on, what actions those clients can take, and what data they can view or update. You also can use load balancers and firewalls to secure your solution.

However, security flaws can occur in any system. Many companies offer a penetration testing service that can help uncover any vulnerabilities in your solution. If you do not have the resource or knowledge to perform penetration testing on your solution, we recommend that you use a third-party penetration testing company.

For more information, see Penetration Testing on Wikipedia

Testing your security

Your Diffusion solution is made up of multiple components. Ensure that you consider and test for potential security problems in all your components and in their interactions.

It is important to design your solution for security before you even begin any development or configuration. For more information about designing a secure solution, see Design Guide on page 26.

> **Note:** Many companies offer a penetration testing service that can help uncover any vulnerabilities in your solution. If you do not have the resource or knowledge to perform penetration testing on your solution, we recommend that you use a third-party penetration testing company.

Consider these aspects of security for your solution.

URL spaces and ports exposed by your load balancer

What routes does your solution offer to connections from outside?

For more information, see Load balancers on page 625.

Connectors

What ports allow connections to the Diffusion server? What kind of connections are these ports configured to allow?

For more information, see Configuring connectors on page 436.

Users and roles on your Diffusion server

How are connections to the Diffusion server authenticated? What roles and permissions are assigned to authenticated connections? How are different parts of your topic tree secured?

For more information, see Role-based authorization on page 138.

Console

Are connections from outside your organization permitted to access the Diffusion console? Which users are assigned the permission to access the console?

For more information, see Diffusion management console on page 538.

Planning for production

The key to a successful production deployment is planning and preparation.

Consider the following questions when planning for production deployment:

Hard launch or soft launch?

In a hard launch, your solution is rolled out to all of your users at the same time. In a soft launch, your solution is rolled out to only a select group of users.

The advantage of a soft launch is that it enables you to trial your new solution with a subset of your users and discover any remaining issues before rolling out to your whole user base.

Will your users experience any down-time?

How will your deployment affect existing users? Will their clients experience a disconnection? Will the deployment of your new solution force them to upgrade their client version before they can continue to use your solution?

Understand what your users will experience during your deployment and what experience you want them to have.

When are you going to roll out to production?

Select a time that fits best with your business. Consider when you have the most users, when you have certain events for which your system needs to be up, and when your team are available to support and troubleshoot the deployment.

Who do you need to notify in advance?

Do you need to notify your users of upcoming down-time? Do you need to notify your user of actions they must take? Do you have any third parties that provide data or services who need to be notified?

How are you deploying your solution to production?

Are you rolling out all of the components of your solution or just changing some of them? What order are you deploying your components in? Are you going to automate all or some of the deployment process?

What is your roll back plan if something unforeseen happens?

Even with the best testing and planning, problems an occur in a production environment. Developing a strategy in advance for handling problems ensures that you can react quickly if problems occur.

Prepare a go-live checklist

After you have considered all aspects of your deployment, we recommend that you create a go-live checklist detailing all of the tasks necessary across your organization in order to go live.

Deploying to your production environment

For the best results, consider automating deployment of your components and configuration.

Automated deployment to your test environment enables you to quickly iterate your development and roll out new changes into testing. Removing the overheads of setting up a test environment by automating the process, gives your team more time to perform testing.

Automated deployment to your production environment helps reduce the risk of human error. By automating all the steps required to deploy your solution to production, you can easily test your deployment process. Automated deployment is quicker than manual deployment and can reduce the amount of down-time that a deployment might cause. Testing your automated deployment process gives you the chance to measure this down-time duration. You can use this information to appropriately set your service-level agreements.

Tuning

Aspects of tuning Diffusion for better performance or resilience

This section covers aspects of configuring Diffusion to achieve higher levels of performance and covers some of the more advanced features which enable users to get more out of Diffusion.

Concurrency

Diffusion is a multi-threaded server and utilizes concurrent processing to achieve maximum performance. Java NIO technology is utilized so that a separate thread is not required for each concurrent connection and very large numbers of concurrent connections can be handled.

Because Diffusion is a multi-threaded environment it is necessary to have an understanding of concurrency issues when configuring Diffusion for best performance.

This section discusses issues of threading and concurrent processing.

User threads

Server-side components using the Diffusion Java API can make use of the Java threads API to schedule tasks for processing of their own in a separate thread of processing.

You can execute any object of a class that implements the `RunnableTask` interface using one of the `ThreadService.schedule` methods. You can to request a one-off execution of a task, periodic execution at a given interval or execution according to a schedule. Periodic processing can be important to publishers that pull data updates from elsewhere.

Such tasks issued using the thread service are executed using threads from the background thread pool.

Alternatively, users can define their own thread pools to use using the thread service and execute tasks using these thread pools.

NIO Threads

Each connector that is configured in `etc/Connectors.xml` comprises a connector thread that listens for incoming socket connections, accepts them and registers them with an acceptor thread that handles any incoming data notifications. Message decodin is run in the inbound thread pool. Connector and acceptor threads are occupied for the minimum amount of time and are completely non-blocking.

Though performance can be improved in extreme case by adjusting the numbers of these NIO threads, no significant processing occurs within them.

Client multiplexers

A client multiplexer is a separate thread which is responsible for processing messages, queuing for clients (conflating if necessary), taking messages from client queues and sending them to the client or clients. A number of these multiplexers can be configured to improve concurrent processing when there are a large number of clients.

The number of multiplexers can be configured. By default, the number of multiplexers is the same as the number of available cores on the host system of the Diffusion server.

Multiplexers typically batch these output messages into output buffers according to the output buffer size configured for the client connectors.

Thread pools

Diffusion maintains a number of configurable thread pools which are used for a number of purposes

For more information, see Thread pools on page 502. Thread pools can also be accessed programmatically using the `ThreadService` class of Diffusion server API. Refer to the API documentation for more information about this.

The various types of thread pools are as follows:

Inbound thread pool

This is used to obtain a thread to process any inbound message received on an NIO connection. The maximum number of threads configured for this pool must cater for the maximum required concurrency for incoming requests.

Diffusion does not maintain a separate thread for each client connection but rather passes each inbound request from a connection to the inbound thread pool for processing.

For example, when a client subscribes, the input processing happens on an inbound thread from the pool, the subscribe method and topic loader methods are run in one of these threads.

Connector inbound thread pools

Individual connectors can configure their own separate inbound thread pool to override the use of the default. This cannot be required if you want different behaviors for each connector or if there are a lot of connectors. Due to locking on the inbound thread pool, you get better performance if each connector to have its own inbound thread pool.

Background thread pool

The background thread pool is used for executing scheduled tasks. These tasks can be issued by Diffusion.

Diffusion uses scheduled tasks for various reasons such as retrying connections. If a Diffusion server cannot connect to another server and there is a retry policy, a scheduled task will be used to retry the connection.

User thread pools

Within the Java threads API user can define thread pools that can be used for multi-threaded processing.

Buffer sizing

There are a number of places within the configuration of Diffusion where buffer sizes must be specified and getting these right can have a significant impact upon performance.

Connector output buffers

An output buffer size must be configured for each connector.

The output buffer size configured for a connector must be at least as large as the largest message that is expected to be sent to any client connecting through that connector. However, the buffer size can be much larger so that the messages can be batched at the server, which improves performance.

Each connected client is assigned a socket buffer of the specified size if possible. A warning is logged if a smaller socket buffer was allocated than requested.

In addition each client multiplexer has a buffer of the configured size (as a multiplexer writes to only one of its clients at any one time). The multiplexer buffer is used to batch messages from the client queue before writing and, if the socket buffer does end up being smaller than the configured buffer and the throughput is high, the allocated socket buffer size might become a bottleneck.

Getting the correct output buffer size is vital. Make this too small and the Diffusion server does not batch and write messages to clients at optimal rates. Make them too big, extra memory is consumed or messages might time out and cause the client connection to be closed.

If the output buffer size is larger than the TCP output buffer size, this can cause problems if the client is slow consuming. If a slow-consuming client does not clear messages from the TCP buffer fast enough, messages on the connector buffer which are waiting to move to the TCP output buffer can time out. You can avoid this problem by setting the TCP output buffers for your operating system and the connector output buffers for your Diffusion server to the same value. You can also increase your message timeout interval.

> **Note:** For maximum performance, ideally all clients configure their input buffer size to match the connector's output buffer size.

Client output buffers

As at the server, the output buffer sizes in use must be configured for a client.

In the Java client this is specified in the ServerDetails object used to make the connection. As the Java client does not buffer messages, this only has to be large enough to cater for the largest message size that is sent to the server.

Connector input buffers

Each connector also specifies an input buffer size.

Input buffers receive messages from clients. This buffer must be as large as the largest message expected. If you specify an input buffer size that is less than the maximum message size, the maximum message size is used as the input buffer size.

This size is also used to allocate a receive socket buffer for the client. The socket buffer allocated might actually be less than requested in which case a warning is logged.

For maximum performance, the size used for this buffer must match up with the output buffer size used by clients.

Client input buffers

Clients must also specify the buffer size to use for input.

In the Java client this is specified in the ServerDetails object used to make the connection.

Matching buffer sizes

For optimal throughput it is desirable to match the size of buffers at each end of every connection. The input buffer size used by clients ideally matches the output buffer size at the connector that they connect to. Also the output buffer size specified by clients must match the input buffer size of the connector they connect to.

Message batching

The size of output buffers configured for a connector can be much larger than the largest expected message size because the server uses these buffers to batch client messages which improves performance. Ideally the buffer size is a multiple of the average message size.

> **Note:** When using for HTTP clients, allow between 20 and 250 bytes extra for control information.

Each client multiplexer assigns an output buffer of each size specified by client connectors. So if there were 3 client connectors, each specifying a different output buffer size, and 2 client multiplexers, each multiplexer assigns 3 different buffers (6 in total).

When a client multiplexer is unable to write the contents of an output buffer to a client in one go, the writing is deferred and the multiplexer takes a copy of the remaining data in the output buffer into its own temporary buffer.

Message sizing

The sizing of messages that are sent to clients is very important to the overall performance and this must be carefully considered within the design of your solution.

Every topic message has a fixed header of 6 bytes. It then has the topic path terminated by one byte, plus any user header information that is also included with the message.

It is important to work out the size of the message so that the connector buffers can be set correctly, otherwise Diffusion is unable to put the messages on the wire quickly enough.

Byte pinching

With any messaging system, the smaller the messages, the lower the latency and the faster the system performs. There is a consultancy exercise that Push Technology performs as a service to analyze the messages and reduce them as much as possible. The following list includes a few of the best practices to use:

- Only send data that is required by the client.
- Look at the data format and strip any fat off the message.
- Is the information being sent a true delta?
- Client side data models

Message encoding

If you are sending large messages, it is worth compressing the messages. This happens only once on the server, and then the clients have the technology to decompress them, this also includes JavaScript clients. If other encoding is used, it is worth bearing in mind the CPU overhead required.

Client queues

A maximum queue depth can be configured for client queues so that clients are closed if their message backlog becomes too large.

The maximum queue depth must be chosen carefully as a large size might lead to excessive memory usage and vulnerability to Denial of Service attacks, whilst a small size can lead to slow clients being disconnected too frequently.

Client queues do not take any memory, as Diffusion uses a Zero Copy paradigm, but there are consequences in setting them too small or too large. If the client queue is set too small, once the client has filled its queue the Diffusion server closes the client.

When considering queue depth take into account the average message size and publication rate. Messages that are held in the client queue are not garbage collected and can get promoted, which increases their impact on GC pressure. If messages in the client queue build up, consider the maximum delay in the context of you application. For example: Assuming 100 bytes is the average message size and the application is publishing an average of 100 messages per second. If the client queue is setup to have a maximum depth of 1000 messages this means we allow messages to build up for a slow client for up to 10 seconds, during this time a slow client is building up a cache of 100,000 bytes of messages to be sent.

> **Note:** It is natural for queues to build up a little with spikes in publication rate or momentary bandwidth limits, but the tolerance to such delays is expressed in the client queue depth and must be considered in that context.

Client multiplexers

Tuning multiplexers for optimal performance

The load of batching, conflating and merging messages being sent from Diffusion to outbound clients is spread across client multiplexers. The number of configured client multiplexers must take into account the expected message load and concurrent client connections. The more clients are assigned to a multiplexer the more load it must contend with.

By default, the number of client multiplexers is equal to the number of cores on the host system of the Diffusion server

A client multiplexer processes all client messages into the client queue. Clients are added to the multiplexers according to a round-robin load balancing policy.

Publishers either broadcast on a topic to all subscribed clients or send clients direct messages. When broadcasting all multiplexers are notified and go on to find all subscribed clients which are assigned to the particular multiplexer. When a message is sent to a particular client only that client's multiplexer is notified. It is more efficient to broadcast than it is to send the same message to a large number of clients by iterating over them.

Client multiplexers are non-blocking, high priority threads so having too many can be detrimental, as they are competing for the same resource (CPU). As a rule of thumb, the number of multiplexers must not exceed the number of available logical cores. If a client multiplexer becomes over-subscribed, message latency can increase. For maximum throughput, the number of multiplexers can be set to the number of available cores, but this configuration is only recommended in the case where other threads are assumed to be mostly idle (for example, little inbound traffic, low publisher overhead).

Client multiplexers performance is influenced by the use of merge and conflation policies as those are executed in the multiplexer thread. It is recommended that conflation policy changes and in particular changes to merge conflation logic be profiled and written with performance in mind. In particular the use of locks or any other blocking code is highly discouraged.

Each multiplexer uses a different buffer for each output buffer size that is specified to any connector. If there were three connectors with different output buffer sizes specified, each multiplexer assigns three different buffers. Each multiplexer might also assign an extra buffer for HTTP use. A larger output buffer enables more efficient batching of messages per write, as large writes are generally more efficient but care must be taken to not overwhelm client connections regularly and causing them to be blocked for any period of time.

When a multiplexer is unable to write a message to a client because the buffer has become full, a selector thread is notified. The selector thread is responsible for watching the client and notifying the multiplexer when it becomes writable. The multiplexer remains responsible for writing the message.

Connectors

You can tune your connectors to handle multiple connections and improve performance.

Configuring multiple connectors

It might be beneficial to configure different connectors for different client types, as their requirements can be different.

Buffers

As Diffusion can have tens of thousands of connections at any one time on a machine it is important to make sure that the buffers are set correctly.

Small buffer sizes increase the number of network operations that must be performed to receive and transmit data, reducing efficiency. However, larger buffer sizes require more server-side memory. The server reserves a buffer for each pending network read or write.

For more information, see Buffer sizing on page 499.

Backlog of incoming connections

By default, the Diffusion server requests that the operating system restricts the maximum number of unaccepted TCP connections to the network port managed by the connector to 1000. Additional clients attempting to connect are refused connection.

This maximum number of unaccepted connections can be configured using the `backlog` element to the connector definition in the `Connectors.xml` configuration file. Diffusion accepts new connections very rapidly, so it is rarely necessary to tune this parameter.

In addition, the number of unaccepted client connections on a socket is further constrained by the system-wide limits set by the operating system that you run the Diffusion server. Ensure that the operating system allows at least as many incoming connections as the Diffusion server.

- On Linux, you can do this by setting the value of `net.core.somaxconn`.

Thread pools

Thread pools are used within Diffusion to optimize the use of threads.

It is important to understand balance when tuning thread usage for a system. There must be sufficient thread resources required but not so many as to starve other parts of the system. At the end of the day there are only so many threads that a system can provide.

In general, when provisioning threads, separate the blocking and non-blocking activities. While it is beneficial to have more threads than cores for blocking tasks it is detrimental to the server if more threads than cores are runnable at any given time.

There are a number of places where thread pools are used within Diffusion. For more information, see Concurrency on page 497.

Configurable properties

The following key values can be configured for a thread pool to influence its behavior and use of resource:

Table 44: Values that can be configured for a thread pool

Property	Usage
Core size	The core number of threads to have running in the thread pool. Whenever a thread is required a new one is created until this number is reached, even if there are idle threads already in the pool. After reaching this number of threads then at least this number of threads is maintained within the pool.
Maximum size	The maximum number of threads that can be created in the thread pool before tasks are queued. If this is specified as 0, the pool is unbounded and so the task queue size value is ignored. Generally an unbounded pool is not recommended as it can potentially consume all machine resources.
Queue size	The pool queue size. When the maximum pool size is reached then tasks are queued. If the value is zero, the queue is unbounded. If not zero then the value must be at least 10 (it is automatically adjusted if it is not).
Keep-alive time	The time limit for which threads can remain idle before being terminated. If there are more than the core number of threads currently in the pool, after waiting this amount of time without processing a task, excess threads are terminated. A value of zero (the default) causes excess threads to terminate immediately after executing tasks.
Notification handler	A thread pool can have a notification handler associated with it to handle certain events relating to the pool. This allows for user written actions to be performed (for example, sending an email) when certain pool events (like too much task queuing) occur. See below for more details.

Property	Usage
Rejection handler	A thread pool can have a rejection handler associated with it to handle a runnable task that has been rejected. This allows user written actions to handle a runnable task that can not be executed by the thread pool. See below for more details.

Notification handler

A thread pool notification handler can be configured to act upon certain thread pool events.

These events are:

Table 45: Events that a thread pool notification handler can act on

Event	Description
Upper threshold reached	A specified upper threshold for the pool has been reached. This means the pool size has reached the specified size. The event is notified only once and is not notified again until the lower threshold reached event has occurred.
Lower threshold reached	A specified lower threshold for the pool has been reached after an upper threshold reached event has been notified. This means the pool size has now shrunk the specified size.
Task rejected	The pool has rejected a task because there are no idle threads available and the task queue has filled. What happens to the rejected task depends upon the type of pool. Typically, the task is run within the thread that passes the task to the pool, which is not desirable. This is why the thread ought to be notified when it occurs. This differs from the rejection handler in that it does not expose the runnable task. This means it can be used only for notification.

The notification handler is a user written class which must implement the `ThreadPoolNotificationHandler` interface in the threads Java API. The name of such a class can be configured for in-bound or out-bound thread pools or for connector thread pools in which case an instance of the class is created (and must have a no arguments constructor) when the thread pool is created.

Rejection handler

A thread pool can have a rejection handler associated with it to handle a runnable task that has been rejected.

Two rejection handlers are provided with Diffusion. These are the `ThreadService.CallerRunsRejectionPolicy` and `ThreadService.AbortRejectionPolicy`.

The `ThreadService.CallerRunsRejectionPolicy` executes the runnable task in the thread that tried to pass the runnable task to the thread service. This can cause inconsistencies and out of order processing.

The `ThreadService.CallerRunsRejectionPolicy` does not execute the task and instead generates an exception.

> **Note:** By default, the thread that tried to pass the runnable task to the thread service blocks until there is space on the thread pool queue.

The rejection handler is a user written class which must implement the `ThreadPoolRejectionHandler` interface in the threads Java API. The name of such a class can be configured for inbound or outbound thread pools or for connector thread pools in which case an instance of the class is created (and must have a no arguments constructor) when the thread pool is created.

Adjusting the configuration

Adjust thread pools gradually. Ideally, duplicate expected maximum loads in test environment. This environment can be used to tune the thread pools to satisfy the load. Tune the thread pools so they are just able to cope with the maximum load, increasing them beyond this might degrade overall performance.

Background thread pool:

In general, the defaults suffice for the tasks assigned to the background thread pool by Diffusion. If you assign tasks to the pool yourself, consider increasing the number of threads.

Inbound thread pool:

This pool is used to handle inbound connections and messages. Increasing the thread pool allows new connections and received messages to be handled over a greater number of threads. However, much of the behavior in this pool can involve locking the clients or parts of the topic tree. This can cause lock contention that delays processing.

Due to the underlying implementation of Java NIO sockets a high rate of threads being added/removed from the incoming thread pool will result in the allocation of off-heap byte buffers. In extreme cases this can result in an out of memory exception being thrown as the server runs out of off heap allocation space.

Session reconnection

You can configure the session reconnection feature by configuring the connectors at the Diffusion server to keep the client session in a disconnected state for a period before closing the session.

When a client detects connection loss, it will automatically attempt to re-establish connection to the server based on the reconnection settings.

Reconnection is enabled by default.

If reconnection is successful, the client session can continue without loss of subscriptions, topic updates, and messages that were queued for it whilst disconnected.

If reconnection is disabled, or fails to re-establish a connection, the client application must create a new session, and re-initialise subscriptions and other application state.

Server configuration

To enable clients to reconnect, connectors must be configured to keep client sessions in the DISCONNECTED state for a period during which the client can reconnect. To do this a reconnection timeout must be specified for the connector.

Specify a reconnection timeout, maximum queue depth, and recovery buffer size by using the `<reconnect>` element in the `etc/Connectors.xml` configuration file.

Reconnection timeout (`keep-alive`)

How long a disconnected client's session remains available on the server before being closed. By default, this is 300 seconds.

Maximum queue depth (`max-depth`)

Optional maximum limit on the number of messages to queue for a disconnected client session. By default, this is the same as the queue depth for a connected client session, which is defined by the queue definitions in `Connectors.xml` and `Server.xml`.

Recovery buffer size (`recovery-buffer-size`)

The maximum number of sent messages to keep in a buffer. These messages can then be recovered on reconnection.

Here is an example connector configuration:

```
<connector>
 ...
 <reconnect>
  <keep-alive>60s</keep-alive>
  <max-depth>1000</max-depth>
  <recovery-buffer-size>64</recovery-buffer-size>
 </reconnect>
 ...
</connector>
```

Using the above example, a client can reconnect to the server through this connector within 60 seconds of becoming disconnected. While the client is disconnected, up to 1000 messages are queued for it. These messages are delivered to the client when it reconnects. A buffer of up to 64 sent messages are retained in the recovery buffer. When a client reconnects, the Diffusion server uses this buffer to re-send any messages that the client has not received.

If a client signals that it wants to disconnect, the client state on the server is removed when the client disconnects. However, in all other circumstances where the client loses connection, the client goes into the DISCONNECTED state, where the subscriptions are retained and messages are queued as normal for the amount of time specified by the reconnection timeout of the connector.

If the server is configured to expect reconnecting clients, sessions that are currently disconnected and might reconnect are excluded from the regular system pings that the server sends to clients.

If the client then reconnects during the period that the session is in DISCONNECTED state, the sending of messages to the client resumes from the point when the failure occurred.

Disabling session reconnection

Session reconnection is on by default. Disabling reconnection can be useful in some circumstances where it is better for a session to fail quickly than for it to be kept alive and await reconnection.

- For example, when you have a control client in the same data center as the Diffusion server, enabling reconnection for the session between the control client and the server can make it harder to diagnose connection problems.

- During development you may want to kill and recompile or reconfigure a client, then restart it. If reconnection is enabled, the original session will not end when the original client is killed. This can lead to problems like preventing the new client from registering as an updater until the reconnection timeout expires.

You can either disable reconnection in the server configuration, or a client can specify that a session should not use reconnection.

Message queue management

When a client session is in DISCONNECTED state, messages for the client continue queuing for the client until the reconnection timeout expires or the client reconnects. This puts an unusual load on the client queue and the facility exists to adjust the maximum client queue depth for the period of disconnection.

This is done by specifying a queue depth which is greater than the normal maximum queue depth. When disconnected, the queue can expand to the higher value. After reconnection occurs and the queue starts to drain, once the queue size goes down to a value of 80% of its previous limit, the maximum queue depth reverts to the normal value.

The queue depth has an effect only if has a value higher than the normal maximum queue depth.

Related concepts

Reconnect to the Diffusion server on page 203
When clients connect to the Diffusion server over unreliable networks these connections can be lost. Clients can attempt to reconnect to the Diffusion server after they lose connection.

Specifying a reconnection strategy on page 206
Reconnection behavior can be configured using custom reconnection strategies.

Client failover

You can configure a client to fail over to another Diffusion server after it loses connection to the Diffusion server it was previously connected to.

Client failover is when a client loses its connection to a server and attempts to connect to a different one. The client is provided with a list of servers. If a client loses its connection to a server it can automatically attempt to connect to the next server in the list. If it fails to connect or loses its connection to that server, it tries the next server on the list. This is referred to as autofailover. Generally the list of servers to connect to must be provided before attempting to make the connection. How the list of servers is provided differs between client APIs and the JavaScript client does not support autofailover but it can be implemented using the callback methods.

Using automatic failover

If a client has an established connection that it loses, autofailover attempts to open a new connection in the next connection in the list. This is not compatible with reconnection because reconnection attempts to preserve the state of the client (the client ID and the subscribed topics). As the new server has no knowledge of the client it is unable to preserve this state. Autofailover must be enabled and a list of servers to connect to provided.

Using load balancing with autofailover

You can enable load balancing in conjunction with autofailover. When load balancing is enabled and a client loses connection, the list of servers is shuffled before the client selects the next server to attempt to connect to.

In Java, for example, you can enable load balancing by using the `setLoadBalancing` method on the `ConnectionDetails` object.

Using server cascading

When a client attempts to place a connection, if the attempt fails, the next server in the list is chosen. Server cascading is similar to autofailover except this logic is applied prior to a connection, whereas autofailover applies once a connection is in place.

In Java, for example, you can enable server cascading by using the `setCascading` method on the `ConnectionDetails` object.

> **Note:** Server cascading is different to protocol cascading, which attempts to connect to the same server using different protocols before a connection has been opened.

Java memory usage

Typically you do not have to tune the Java VM's use of memory. However, in certain conditions, consider using runtime options to change the default behavior.

If you use SSL-offloading at Diffusion

If your clients make secure connections to the Diffusion server and these connections are SSL offloaded at the Diffusion server, ensure that you tune the following runtime options:

-Xmx

Sets the maximum heap size.

-XX:MaxDirectMemorySize

Sets the maximum total size (in bytes) of direct-buffer allocations. By default, the JVM chooses the size for direct-buffer allocations automatically.

Diffusion uses direct memory to offload SSL connections.

Ensure that the combined total of these two values does not exceed 80% of the RAM available on your system.

When terminating SSL connections at Diffusion, Java can consume significant CPU resource in encryption library code. Run the Diffusion server on Java 8 update 121 or later to take advantage of flags which enable optimizations which significantly reduce CPU utilization when the Diffusion server receives a sustained and high client session connection rate.

Platform-specific issues

To run Diffusion it might be necessary to increase the number of sockets and reduce timewait.

It might also be necessary to increase the number of open files that is allowed on UNIX or Linux systems

Socket issues

To fix these problems, complete the following steps based on platform.

Windows

Setting values on Windows

Setting TCP timed wait

This parameter determines the length of time that a connection stays in the TIME_WAIT state when it is closed. When a connection is in the TIME_WAIT state, the socket pair cannot be reused. This is also known as the 2MSL state because the value is twice the maximum segment lifetime on the network. See RFC 793 for further details.

Add TcpTimedWaitDelay registry values as a workaround. You can set these values through REGEDIT command.

Set TcpTimedWaitDelay to 30:

1. Select **Start** > **Run**.
2. In the available field, enter `regedit`.
3. Go to the key directory file: `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Tcpip/Parameters/TcpTimedWaitDelay`. The value type is `REG_DWORD`.
4. Double-click **TcpTimedWaitDelay**.
5. Select **Decimal**.
6. Type `30` in the **Value** data field. The default value for this field is 0xF0 (240 decimal). The valid range is 30-300 (decimal).

Setting MaxUserPort

This parameter controls the maximum port number used when an application requests any available user port from the system. Normally, short-lived ports are allocated in the range from 1024 through 5000. Setting this parameter to a value outside of the valid range causes the nearest valid value to be used (5000 or 65534).

Add MaxUserPort registry values as a workaround. You can set these values through REGEDIT command.

Set the MaxUserPort to 65534

1. Select **Start** > **Run**.
2. In the available field, enter `regedit`.
3. Go to the key directory file: `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Tcpip/Parameters/MaxUserPort`. The value type is `REG_DWORD`.
4. Double-click **MaxUserPort**.
5. Select **Decimal**.
6. Type `65534` in the **Value** data field. The default value for this field is 0x1388 (5000 decimal). The valid range is 5000 – 65534(decimal).

Linux

Configuring sockets values on Linux

Decrease the time wait before closing the sockets by entering:

```
# echo 3 > /proc/sys/net/ipv4/tcp_fin_timeout
```

Sometimes systems are now configured to prevent one from using a large number of ports, check the port range and modify if required.

```
# cat /proc/sys/net/ipv4/ip_local_port_range
```

This can be increased by issuing the following command

```
# echo "1025 65535" > /proc/sys/net/ipv4/ip_local_port_range
```

To have these new values take effect you might have to do (as root)

```
# /etc/rc.d/init.d/network restart
```

If you want these new values to survive across reboots you can at them to `/etc/sysctl.conf`.

```
# Allowed local port range
net.ipv4.ip_local_port_range = 1025 65535
# net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_fin_timeout = 3
```

UNIX

Increasing the number of files a process on a UNIX system can open also increases the number of sockets that process can open. The operating system uses file descriptors to handle filesystem files as well as pseudo files, such as connections and sockets.

You need the following number of sockets for each client connection:

- WS — one socket per connection
- HTTP Polling — two sockets per connection

Use the `ulimit` command to increase the number of open files. You can do this in one of the following ways:

- As a global setting.

  This can be set by your network administrator.
- In the start script for Diffusion.

  Edit the *`diffusion_installation`*`/bin/diffusion.sh` file to add the following line at the start:

  ```
  ulimit -n open_files
  ```

  Where the value of *open_files* is any suitable integer value, for example 8192.

You can use the `java.lang:type=OperatingSystem` MBean to inspect the number of files on your UNIX operating system. See the following properties:

`MaxFileDescriptorCount`
: The total number of files that a process on a UNIX system can open. This is the number that you can set with `ulimit -n`.

`OpenFileDescriptorCount`
: The number of files that are currently open.

The difference between these values is the number of files you have available to use for sockets.

Managing and monitoring your running Diffusion server

This section discusses how to manage your Diffusion server and system as a whole.

We recommend that you actively monitor the health of your Diffusion server to pre-empt failures and to minimize unplanned downtime.

You can monitor your Diffusion server using the tools listed in this section.

JMX

You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

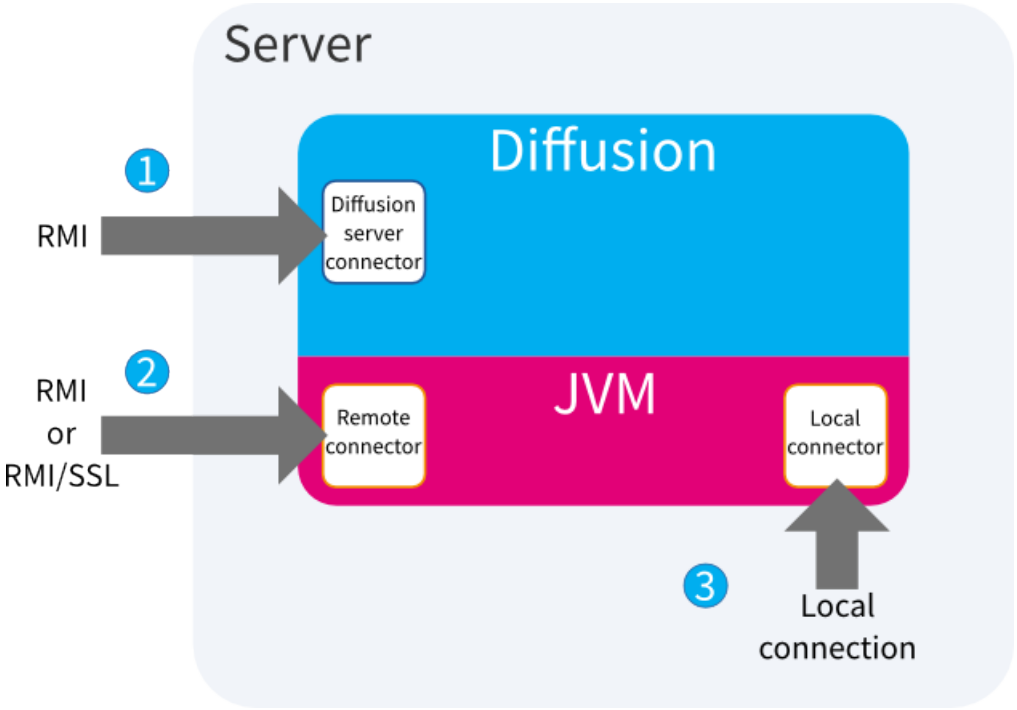


Figure 22: Connecting to Diffusion JMX

The following methods of connecting to Diffusion JMX are available:

1. **Recommended:** Through the RMI JMX connector server provided by the Diffusion server.

   This feature is integrated with Diffusion security, enabling you to use roles and permissions to control access to the MBeans. However, this connection does not use SSL.

   For more information, see Configuring the Diffusion JMX connector server on page 462.
2. Through the RMI JMX connector server provided by JVM that runs Diffusion.

   You can use SSL to make a secure connection. However, the JVM does not use Diffusion security. You must add additional configuration to your JVM to control access to the MBeans.

   For more information, see Configuring a remote JMX server connector on page 463.
3. Through the local JMX connector server provided by JVM that runs Diffusion.

   You make this connection from the server that Diffusion runs on. However, the JVM does not use Diffusion security. You must add additional configuration to your JVM to control access to the MBeans.

   For more information, see Configuring a local JMX connector server on page 464.

Related tasks

Configuring the Diffusion JMX connector server on page 462
Connect to JMX through the Diffusion connector server. This connector server is integrated with the Diffusion server and enables you to use role-based access control to define how connecting users can use the MBeans.

Configuring a local JMX connector server on page 464

Connect to JMX through a local connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Configuring a remote JMX server connector on page 463
Connect to JMX through a remote connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Related reference

Metrics on page 531
Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Diffusion management console on page 538
A web console for managing the Diffusion server.

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Integration with Splunk on page 615
How to achieve basic integration between Diffusion and the Splunk™ analysis and monitoring application

Management.xml on page 465
This file specifies the schema for the management properties that enable JMX access over an RMI JMXConnectorServer.

Using Java VisualVM

You can manage Diffusion using the JMX system management console Java VisualVM.

Java VisualVM is usually installed with JDK's but can be downloaded from https://visualvm.dev.java.net/.

Connecting to the Diffusion connector server

1. Start Java VisualVM.
2. Right-click on the **Remote** section of the **Applications** panel and select **Add Remote Host**.
3. In the **Host name** field, type the host name or IP address of the server where Diffusion is running. Click **OK**.
4. In the **Applications** panel, right-click on the name of the server where Diffusion is running. Select **Add JMX Connection**.
5. In the **Connection** field, enter the host name and RMI registry port for the Diffusion server.
6. Select **Use security credentials** and enter the username and password of a principal that you have configured to be able to use JMX. For more information, see Configuring the Diffusion JMX connector server on page 462. Click OK.

Information about the Diffusion process is displayed in the main panel.

Connecting to the JVM remote connector server

Note: We recommend you use the Diffusion connector server to access the JMX service.

1. Start Java VisualVM.
2. Right-click on the **Remote** section of the **Applications** panel and select **Add Remote Host**.

3. In the **Host name** field, type the host name or IP address of the server where Diffusion is running. Click **OK**.
4. In the **Applications** panel, right-click on the name of the server where Diffusion is running. Select **Add JMX Connection**.
5. In the **Connection** field, enter the host name and RMI registry port for the Diffusion server.
6. Select **Use security credentials** and enter the username and password of a user that you have configured in the JVM. For more information, see Configuring a remote JMX server connector on page 463. Click OK.

Information about the Diffusion process is displayed in the main panel.

Connecting to the JVM local connector server

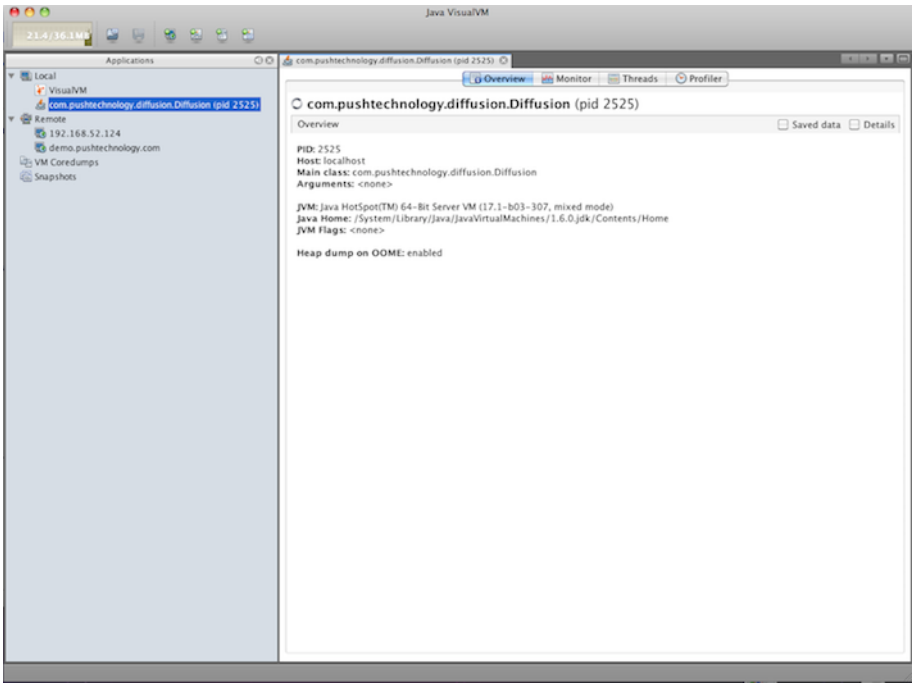


Figure 23: Java VisualVM: Overview tab

Note: We recommend you use the Diffusion connector server to access the JMX service.

1. Start Java VisualVM.
2. From the **Local** section of the **Applications** panel, select the Diffusion process, `com.pushtechnology.diffusion.Diffusion`.
3. Right-click `com.pushtechnology.diffusion.Diffusion` and select **Open**.

Information about the Diffusion process is displayed in the main panel.

Once connected to JMX, several aspects of the system are available to monitor and tune. For more information, see the Java VisualVM documentation: http://visualvm.java.net/docindex.html.

Related tasks

Configuring the Diffusion JMX connector server on page 462
Connect to JMX through the Diffusion connector server. This connector server is integrated with the Diffusion server and enables you to use role-based access control to define how connecting users can use the MBeans.

Configuring a local JMX connector server on page 464

Connect to JMX through a local connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Configuring a remote JMX server connector on page 463
Connect to JMX through a remote connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Using JConsole

You can manage Diffusion using the JMX system management console JConsole.

Connecting to the Diffusion connector server

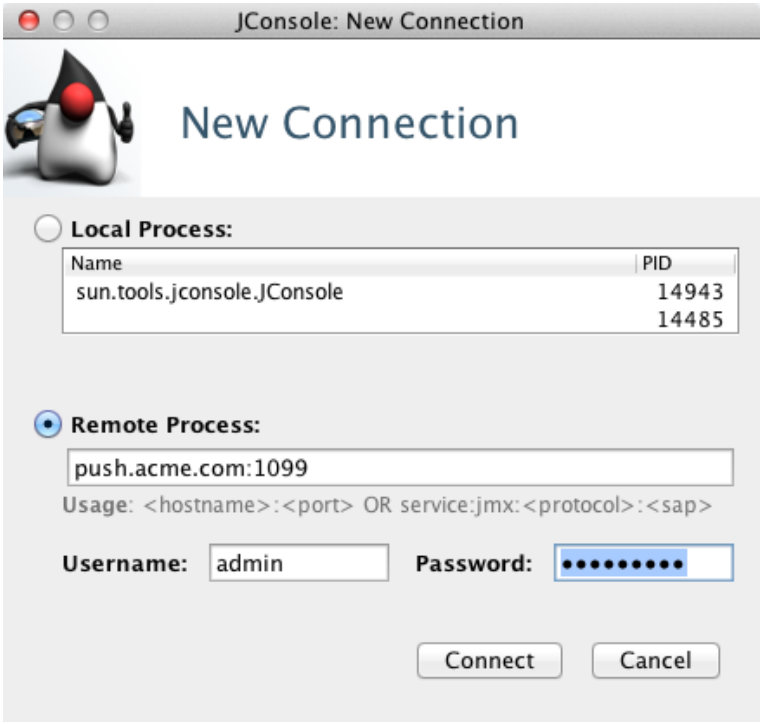


Figure 24: JConsole New Connection dialog: Remote Process

In the **Remote Process** section of JConsole's **New Connection** dialog, enter the following information:

- The host name and RMI registry port of the Diffusion connector server. The default RMI registry port is 1099.
- The username and password of a principal that you have configured to be able to use MBeans. For more information, see Configuring the Diffusion JMX connector server on page 462

Connecting to the JVM remote connector server

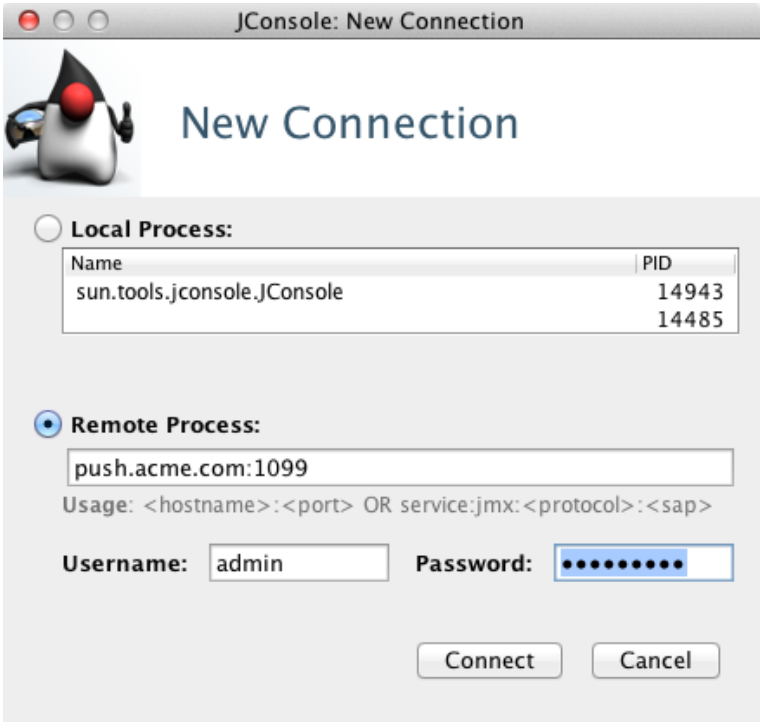


Figure 25: JConsole New Connection dialog: Remote Process

Note: We recommend you use the Diffusion connector server to access the JMX service.

In the **Remote Process** section of JConsole's **New Connection** dialog, enter the following information:

- The host name and RMI port of the Diffusion connector server. The default port is 1099.
- A username and password that you have configured in the JVM to be able to connect to the JMX service. For more information, see Configuring a remote JMX server connector on page 463

Connecting to the JMX service

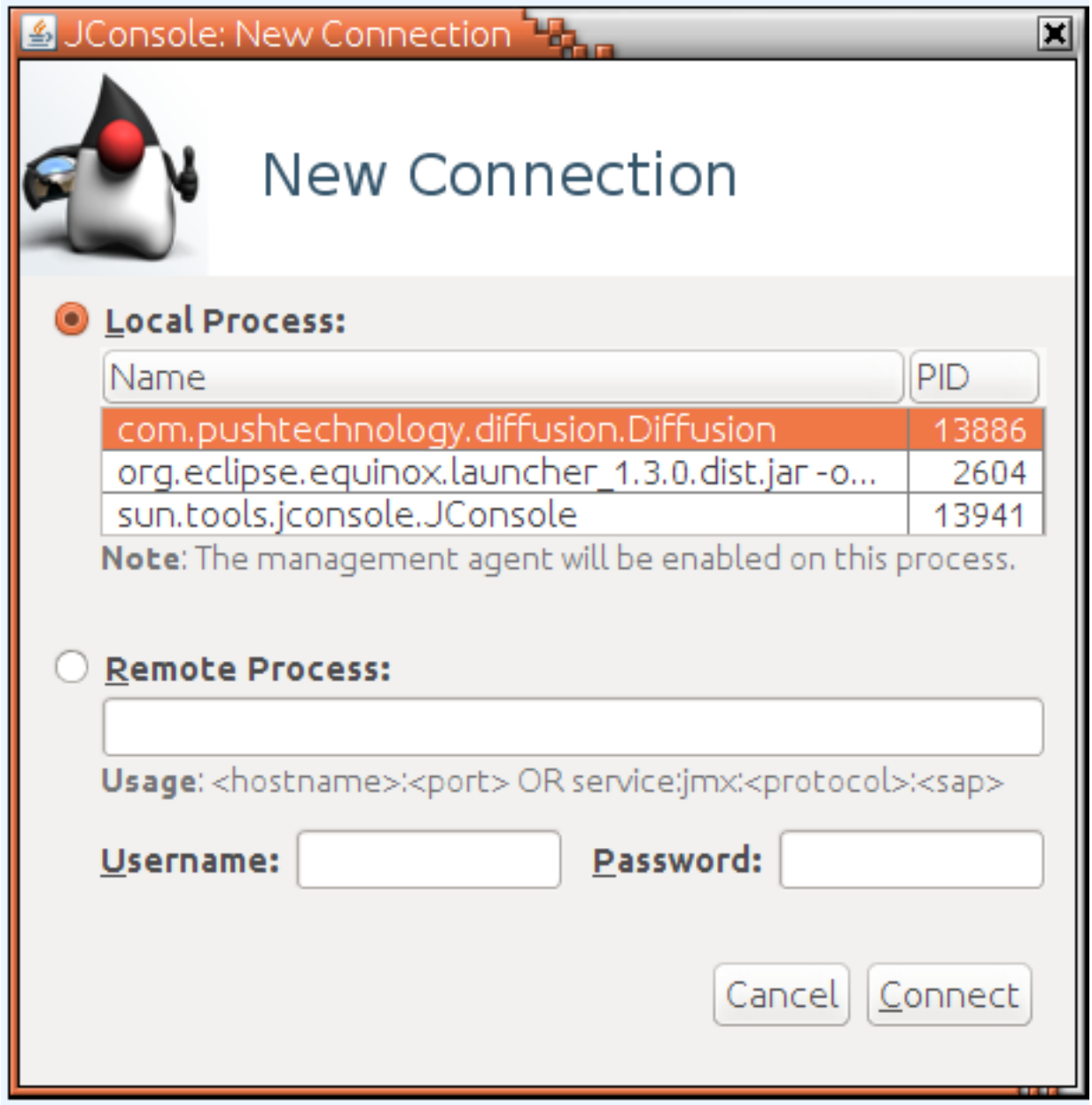


Figure 26: JConsole New Connection dialog: Local Process

Note: We recommend you use the Diffusion connector server to access the JMX service.

In the **Local Process** section of JConsole's **New Connection** dialog, select the Diffusion process, `com.pushtechnology.diffusion.Diffusion`.

Once connected to JMX, several aspects of the system are available to monitor and tune. For more information, see the JConsole documentation: https://docs.oracle.com/javase/8/docs/technotes/guides/management/jconsole.html.

Related tasks

Configuring the Diffusion JMX connector server on page 462

Connect to JMX through the Diffusion connector server. This connector server is integrated with the Diffusion server and enables you to use role-based access control to define how connecting users can use the MBeans.

Configuring a local JMX connector server on page 464
Connect to JMX through a local connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Configuring a remote JMX server connector on page 463
Connect to JMX through a remote connector to the JVM that runs the Diffusion. This connector is not integrated with the Diffusion server security and you must configure additional security in the JVM.

Detecting deadlocks with JConsole

To check if your publisher is experiencing a deadlock, you can use JConsole to inspect the threads.

Procedure

1. Connect JConsole to the JMX service on the Diffusion server.

   For more information, see Using JConsole on page 514.
2. Go to the **Threads** tab.

   This tab provides information about thread use: number of threads, a list of active threads, and details for a selected thread.
3. On the **Threads** tab, click the **Detect Deadlock** button.

   If deadlocks are present, new tabs open next to the **Threads** tab. You can use the information in these tabs to diagnose any deadlocks.

What to do next

For more information, see http://docs.oracle.com/javase/7/docs/technotes/guides/management/jconsole.html.

MBeans

Diffusion registers MBeans with the JMX service for many of its principal features.

Annotations on each of the MBeans employed are used to produce the following pages in this manual as well as feeding JMX clients with descriptive information. MBeans, attributes and operations have descriptions; operation arguments have names; operations also have JMX impact information.

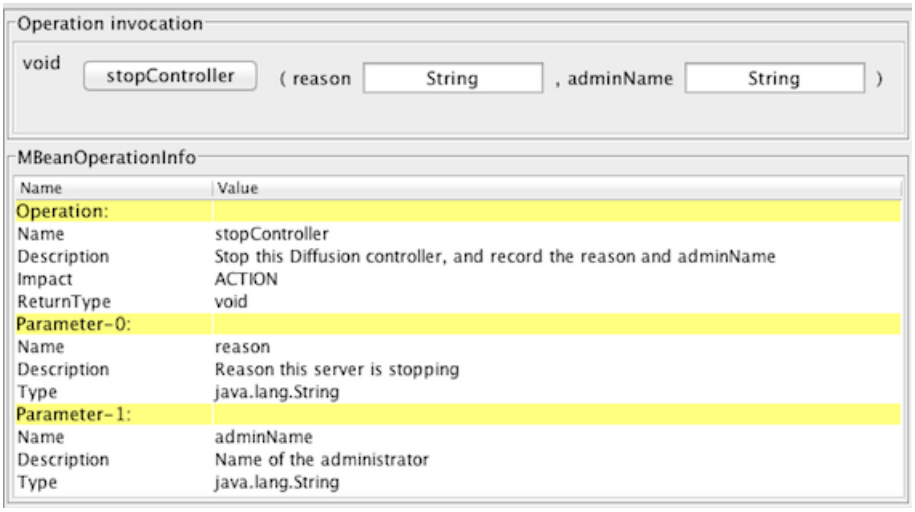


Figure 27: The server MBean stopController operation showing in JConsole

ClientStatistics

Monitoring interface to the client session statistics MBean

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=ClientStatistics,server="server_name"`

Attributes

clientOutputFrequency	long	read-write	Statistics output frequency in milliseconds
clientResetFrequency	long	read-write	The frequency at which the counters are reset
concurrentClientCount	int	read	The current client session count
connectionCounts	Map	read	The current client session count, broken down by client type
maximumConcurrentClientCount	int	read	The maximum number of concurrent client sessions
maximumDailyClientCount	int	read	The count of client sessions started in a day

Connector

Management interface to a connector

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=Connector,name="name",server="server_name"`

Attributes

keepAliveQueueMaximumDepth	int	read-write	The maximum queue depth used for clients in the keep-alive state
keepAliveTime	long	read-write	The time in milliseconds that a unexpectedly disconnected client is kept alive before closing
numberOfAcceptors	int	read	The number of acceptors
queueDefinition	String	read-write	The queue definition
totalNumberOfConnections	long	read	The number of connections accepted since the connector was started
uptime	String	read	The time this connector has been running as a formatted string, or 0 if the connector is not running
uptimeMillis	long	read	The time this connector has been running in milliseconds, or 0 if the connector is not running

Operations

remove	void	0	ACTION	Remove the connector. It will not be possible to restart the connector again (until system restart).

start	void	0	ACTION	Start the connector

stop	void	0	ACTION	Stop the connector. Allows it to be restarted.

Log

Management interface for Log Definition

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=Log,name="name",server="server_name"`

Attributes

description	LogDescription	read	The LogDescription for this log

filename	String	read	The fully qualified filename of this log
logLevel	String	read-write	The current log level as a string

LogMetrics

Log metrics.

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=LogMetrics,server="server_name",code="PUSH-xxxxx",level="level"`

Attributes

count	long	read	Number of log events.

MetricCollectors

Metric collectors.

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=MetricCollectors,server="server_name"`

Attributes

sessionMetricCollectors	List	read	The session metric collectors.
topicMetricCollectors	List	read	The topic metric collectors.

Operations

putSessionMetricCollector	Void	4	ACTION	Add a session metric collector, replacing any with the same name.

name	String	Metric collector name.
exportToPrometheus	boolean	Whether the metric collector should be exposed through the Prometheus gateway.
removeMetricsWithNoMatchingSessions	boolean	Whether metrics should be removed when there are no matching sessions.
sessionFilter	String	Session filter used to select the sessions to collect.

putSessionMetricCollector	void	5	ACTION	Add a session metric collector, replacing any with the same name.

name	String	Metric collector name.
exportToPrometheus	boolean	Whether the metric collector should be exposed through the Prometheus gateway.
removeMetricsWithNoMatchingSessions	boolean	Whether metrics should be removed when there are no matching sessions.
sessionFilter	String	Session filter used to select the sessions to collect.
groupByProperty	String	Session property used to partition the collected metrics.

putSessionMetricCollector	void	6	ACTION	Add a session metric collector, replacing any with the same name.

name	String	Metric collector name.
exportToPrometheus	boolean	Whether the metric collector should be exposed through the Prometheus gateway.
removeMetricsWithNoMatchingSessions	boolean	Whether metrics should be removed when there are no matching sessions.
sessionFilter	String	Session filter used to select the sessions to collect.
groupByProperty	String	Session property used to partition the results.
groupByProperty2	String	Session property used to further partition the results.

putTopicMetricCollector	void	4	ACTION	Add a topic metric collector, replacing any with the same name.

name	String	Metric collector name.
exportToPrometheus	boolean	Whether the metric collector should be exposed through the Prometheus gateway.
topicSelector	String	Topic selector used to select the topics to collect.
groupByTopicType	boolean	Whether the collected metrics should be partitioned by topic type.

removeSessionCollector	void	1	ACTION	Remove the sesion metric collector with the given name.

name	String	Metric collector name.

removeTopicCollector	void	1	ACTION	Remove the topic metric collector with the given name.

name	String	Metric collector name.

Multiplexer

Management interface to a multiplexer

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=Multiplexer,name="name",server="server_name"`

Attributes

latencyWarningTime	long	read	The latency threshold of this multiplexer, after which notifications will be sent
name	String	read	The Multiplexer name
numberOfClients	int	read	The current number of clients assigned to multiplexer
pendingEvents	int	read	The number of operations pending on the multiplexer queue

Operations

diagnosticReport	String	0	UNKNOWN	Generate a diagnostic report describing the state of this multiplexer

startEventDiagnosticRecording	void	1	UNKNOWN	Start an event diagnostic recording or change the end time if a recording is already in progress

duration	long	Record events for this number of milliseconds

stopEventDiagnosticRecording	void	0	UNKNOWN	Stop an event diagnostic recording. The report will be produced to the server log.

Notifications

javax.management.Notification	Published in case of multiplexer latency (deprecated)

MultiplexerManager

Management interface to the multiplexer manager

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=MultiplexerManager,name="name",server="server_n`

Attributes

numberOfMultiplexers	int	read	The number of multiplexers

Operations

diagnosticReport	String	0	UNKNOWN	Generate a diagnostic report for each multiplexer

startEventDiagnosticRecording	void	1	UNKNOWN	Start an event diagnostic recording for each multiplexer, or change the end time if a recording is already in progress

duration	long	Record events for this number of milliseconds

stopEventDiagnosticRecording	void	0	UNKNOWN	Stop all event diagnostic recording. Reports will be produced to the server log.

NetworkMetrics

Network metrics.

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=NetworkMetrics,server="server_name"`

Attributes

inboundBytes	long	read	Data received from the network in bytes.
outboundBytes	long	read	Data sent to the network in bytes.

Server

Diffusion Server

Object name format

The objectName for MBeans of this type is of the following form:
`com.pushtechnology.diffusion:type=Server,server="server_name"`

Attributes

buildDate	String	read	Build date and time of the Diffusion software
freeMemory	long	read	Free memory available in the Java heap
licenseExpiryDate	Date	read	License expiry date
licenseProperties	Map	read	License properties
maxMemory	long	read	Maximum Java heap memory that can be allocated
numberOfTopics	long	read	Number of topics hosted by this server
release	String	read	Diffusion version, for example, 6.2.1_01
sessionLocks	Map	read	Allocated session locks
startDate	Date	read	Date and time at which this server was started
startDateMillis	long	read	Time at which this server was started, as milliseconds since the epoch
timeZone	String	read	Time zone this server is using
totalMemory	long	read	Total memory allocated to the Java heap
uptime	String	read	Time this server has been running, as a formatted string. For example, "3 hours 4 minutes 23 seconds"

uptimeMillis	long	read	Time this server has been running, in milliseconds
usedPhysicalMemorySize	long	read	Used physical memory, in bytes
usedSwapSpaceSize	long	read	Used swap space, in bytes
userDirectory	String	read	Directory in which this server was started
userName	String	read	User account under which this server is running

Operations

getSessionLock	SessionLockBean	1	INFO	Query a session lock by name. Returns null if the lock is not allocated.

lockName	String	Session lock name

checkLicense	void	0	ACTION	Deprecated - has no effect

stopController	void	0	ACTION	Stop this server

stopController	void	2	ACTION	Stop this server, and record the reason and administrator name

reason	String	Reason this server is stopping
adminName	String	Name of the administrator

ServiceMetrics

Service metrics.

Object name format

The objectName for MBeans of this type is of the following form:

```
com.pushtechnology.diffusion:type=ServiceMetrics,server="server_name",directon="i
out",service="SERVICE_NAME"
```

Attributes

errors	long	read	Number of service errors.
requests	long	read	Number of service requests.
responses	long	read	Number of service responses.

SessionMetrics

Session metrics.

Object name format

The objectName for MBeans of this type is of the following form:

```
com.pushtechnology.diffusion:type=SessionMetrics,server="server_name"[,name="metr
collector name"]
```

Attributes

connectedSessions	long	read	Current number of connected sessions.
inboundBytes	long	read	Session data received from the network in bytes.
inboundMessages	long	read	Session messages received from the network.
openSessions	long	read	Current number of sessions.
outboundBytes	long	read	Session data sent to the network in bytes.
outboundMessages	long	read	Session messages sent to the network.
peakSessions	long	read	Peak number of connected sessions.
totalSessions	long	read	Total sessions opened since the server started.

TopicMetrics

Topic metrics.

Object name format

The objectName for MBeans of this type is of the following form:

```
com.pushtechnology.diffusion:type=TopicMetrics,server="server_name"[,name="metric
collector name"]
```

Attributes

bytes	long	read	Stored data in bytes.
count	long	read	Current number of topics.

deltaBytes	long	read	Each change of value increases this metric by the size of the delta representing the difference between the previous and new values.
deltaCompressionRatio	double	read	A value between 0 and 1 representing the achievable delta compression. The calculation is: valueBytes == 0 ? 0 : 1 - deltaBytes/valueBytes.
deltaUpdates	long	read	Number of updates providing a delta.
subscriberUpdateBytes	long	read	Value and delta update bytes sent to subscribers, before message compression.
subscriberUpdateCompressedBytes	long	read	Value and delta update bytes sent to subcribers, after message compression.
subscriberUpdates	long	read	Number of updates sent to subscribers.
subscribers	long	read	Number of sessions with direct subscriptions.
subscriptions	long	read	Number of direct subscriptions.
total	long	read	Total topics created since the server started.
valueBytes	long	read	Each change of value increases this metric by the size of the new value.
valueUpdates	long	read	Number of updates providing a full value.

VirtualHost

HTTP virtual host management interface

Object name format

The objectName for MBeans of this type is of the following form:

```
com.pushtechnology.diffusion:type=VirtualHost,name="name",server="server_name",we
```

Attributes

cacheSizeBytes	int	read	The cache size in bytes
cacheSizeEntries	int	read	The number of entries in the cache
aliasFile	String	read	the alias file name
compressionThreshold	int	read	the compression threshold
debug	boolean	read	true if debug is set
documentRoot	String	read	the document root directory
errorPage	String	read	the error-page file name

fileServiceName	String	read	the file service name
homePage	String	read	the home-page file name
host	String	read	the host name
minify	boolean	read	true if the minify property is set
name	String	read	the virtual host name
numberOfRequests	int	read	number of requests actioned since service was started
static	boolean	read	true if static
webServerName	String	read	the web server name

Operations

startService	void	0	ACTION	Restart a previously stopped virtual host

stopService	void	0	ACTION	Stops the virtual host from processing requests

clearCache	void	0	ACTION	Clear the cache of all entries

The JMX adapter

The JMX adapter reflects JMX MBeans and their properties and notifications as topics.

The JMX adapter is packaged in the Diffusion publisher. The Diffusion publisher must be running for the JMX adapter be enabled.

You can configure the adapter to reflect the state of JMX MBeans and MXBeans as topics. These MBeans can be built-in, Diffusion, or third-party in origin.

The following aspects of the JMX adapter can be configured:

- Whether it is enabled or disabled.

  By default, the adapter is enabled.
- Which MBeans are reflected as topics.

  By default, all Diffusion MBeans, `java.nio:*`, `java.lang:*`, and `java.util.logging:*` are reflected as topics.
- How often the data on those topics is refreshed.

  By default, the topics are refreshed every 3 seconds.

For more information about configuring the JMX adapter, see

Many statistics are available as MBean properties, for example, CPU load, OS version, number of file-descriptors, and threads. Making these statistics available as topics to Diffusion clients makes possible the implementation of system monitoring solutions to the web, and all other Diffusion platforms.

Note: Publishing MBean data to topics can constitute a security risk. Ensure that crucial information about your Diffusion server is protected by permissions.

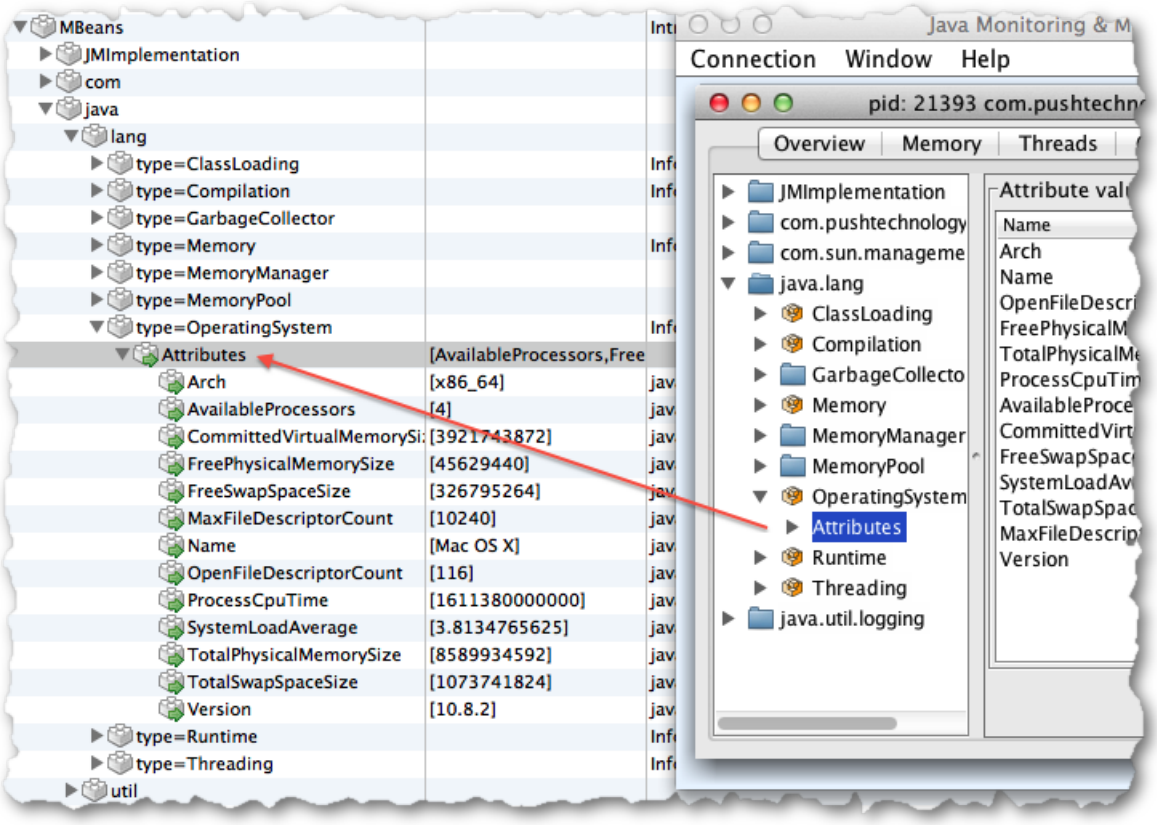


Figure 28: Reflecting MBeans as topics

MBean notifications are also available as topics. Whenever a notification is thrown and the matching topic is subscribed and a message holding a number of key attributes is published to it.

Table 46: Notifications as topics

Record starting ...	Holding
message	`javax.management.Notification.getMessage()`
sequenceNumber	`javax.management.Notification,getSequenceNumb`
timeStamp	`javax.management.Notification.getTimeStamp()`
userData	`javax.management.Notification.getUserData()` if present
source	`javax.management.Notification.getSource()`

The JMX Adapter is itself an MBean with object-name `com.pushtechnology.diffusion:name=JMXAdapter`, which exposes the polling frequency in milliseconds as attribute 'UpdateFrequency'. A value less than or equal to zero prevents polling.

MXBeans versus Simple MBeans

The JMX adapter caters for both MXBeans and simpler MBeans. All MBean attributes are serialized as strings when converted to topics, this might be impractical if a solution returns an object or an array of objects. MXBean attributes with ArrayType and CompositeType types are treated differently.

- CompositeType Fields within the composite attribute are mapped to discrete topics.

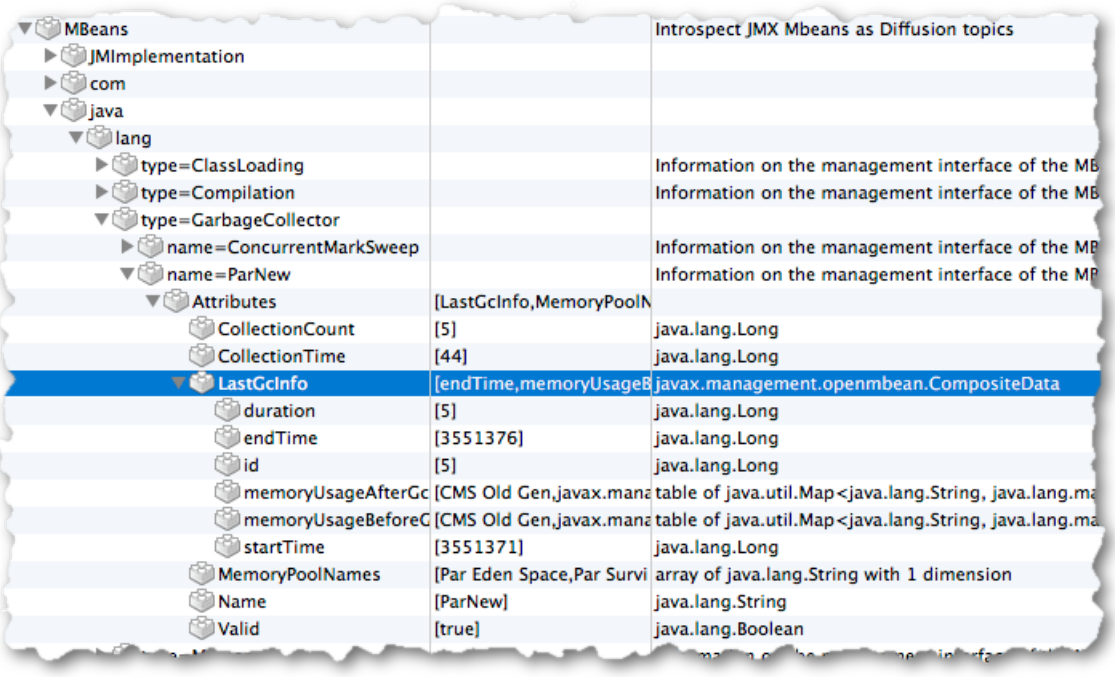


Figure 29: Showing a composite attribute as a topic nest

- ArrayType One dimensional arrays are presented as a single record with many values. Two dimensional arrays are not supported. ArrayType attributes holding attributes that are not SimpleType are not supported (for example, an ArrayType attribute holding Composite or ArrayType values)

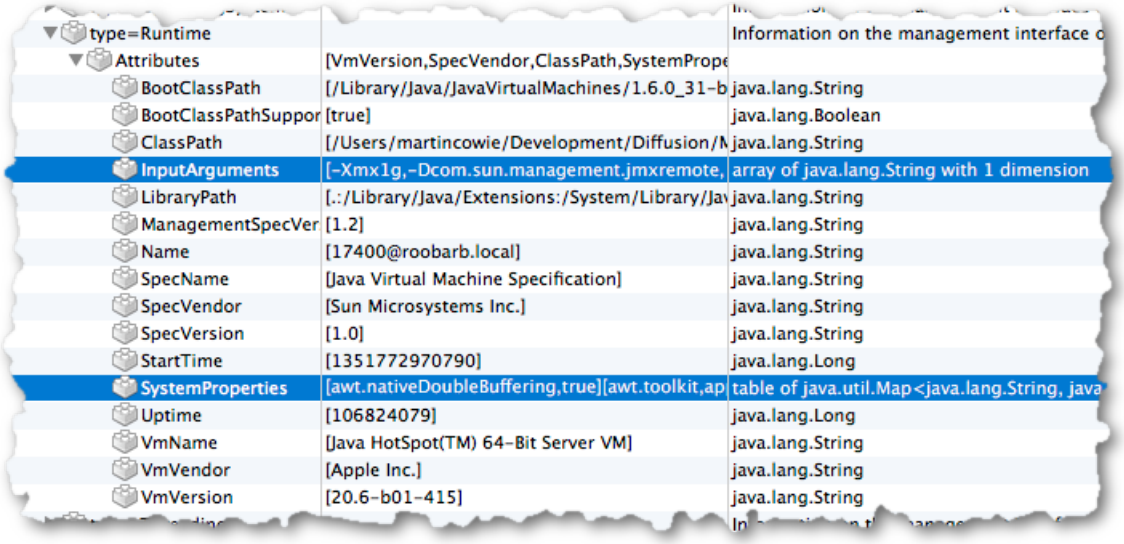


Figure 30: Topics reflecting an ArrayType MXBean attributes

Metrics

Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Methods of accessing metrics

There are multiple ways to access the metrics. As of Diffusion 6.3, the same information is available through each access method.

Note: In previous versions of Diffusion, metrics were sometimes referred to as "statistics".

Web console metrics

The metrics are available through the Diffusion web console. This is the most convenient way to access metrics for development and testing purposes, but does not support aggregating metrics across multiple servers or recording and retrieving historical data. JMX or Prometheus access are more suitable for production systems.

MBeans for JMX

Diffusion registers MBeans with the JMX service. This enables monitoring of the metrics using the JMX tools that are available from a range of vendors.

Prometheus

Diffusion provides endpoints for the Prometheus monitoring system. To use Prometheus, your Diffusion server needs to have a Commercial with Scale & Availability license, or an evaluation license such as the Community Evaluation license. See License types on page 37 for more information.

Accessing metrics

The metrics can be accessed in the following recommended ways:

- As MBeans, using a JMX tool, such as VisualVM or JConsole. See the table below for MBean interfaces. For more information, see Using Java VisualVM on page 512 or Using JConsole on page 514.

- Using the Diffusion management console. For more information, see Diffusion management console on page 538.
- As Prometheus endpoints at `http://localhost:8080/metrics`, provided you have a suitable license. If not accessing from the same machine as the Diffusion server, replace `localhost` with the IP address or hostname.

Collecting custom metrics using metric collectors

A metric collector is a way to collect metrics for a particular set of topics or sessions, configured by you.

You can use the Diffusion web console or JMX to define metric collectors. See Configuring metrics on page 535 for details.

Collected metrics are published to the console, JMX and optionally via Prometheus.

Counters and gauges

Metrics are divided into counters and gauges.

Counter metric

A counter is a cumulative metric, which reports a value since the server was started. A counter metric will always go up over a server's lifetime. For example, the total number of bytes received by the server is a counter.

Gauge metric

A gauge is a metric which reports the current value of a metric. A gauge value can go up or down. For example, the number of connected sessions is a gauge.

Built-in metrics

This section describes the built-in metrics that are always available, aside from any metric collectors you may have created.

Metrics are not persisted between server restarts. Restarting the server will set all counter metrics back to zero.

The following is a list of all the top level statistics and their attributes.

Table 47: Metrics provided by Diffusion

Metric name	Type	Description	Prometheus export
Log metrics	LogMetrics on page 520 MBean		
count	Counter	Number of log events for a given ID code and severity level (levels are `error`, `warn`, `info`, `debug`, `trace`).	diffusion_log_events_count{code="P
Network metrics	NetworkMetrics on page 524 MBean		
inbound_bytes	Counter	Data received from the network in bytes.	diffusion_network_inbound_bytes
outbound_bytes	Counter	Data sent to the network in bytes.	diffusion_network_outbound_bytes
Session metrics	SessionMetrics on page 526 MBean		

Metric name	Type	Description	Prometheus export
connected	Gauge	Number of connected sessions.	diffusion_sessions_connected
inbound_bytes	Counter	Session data received from the network in bytes.	diffusion_sessions_inbound_bytes
inbound_messages	Counter	Session data received from the network in messages.	diffusion_sessions_inbound_messag
open	Gauge	Number of open sessions.	diffusion_sessions_open
outbound_bytes	Counter	Session data sent to the network in bytes.	diffusion_sessions_outbound_bytes
outbound_messages	Counter	Session data sent to the network in messages.	diffusion_sessions_outbound_messa
peak	Gauge	Peak number of sessions.	diffusion_sessions_peak
total	Counter	Total sessions opened.	diffusion_sessions_total
Topic metrics	TopicMetrics on page 526 MBean		
count	Gauge	Current number of topics.	diffusion_topics_count
total	Counter	Total number of topics.	diffusion_topics_total
bytes	Gauge	The value data stored by the topics, in bytes.	diffusion_topics_bytes
subscriptions	Gauge	Number of direct subscriptions to the topics.	diffusion_topics_subscriptions
subscribers	Gauge	Number of sessions subscribed to one or more topics.	diffusion_topics_subscribers
subscriber_updates	Counter	Number of updates sent to subscribers.	diffusion_topics_ subscriber_updates
subscriber_update_bytes	Counter	Data sent to subscribers, before message compression, in bytes.	diffusion_topics_ subscriber_update_bytes
subscriber_update_ compressed_bytes	Counter	Data sent to subscribers, after message compression, in bytes.	diffusion_topics_subscriber _update_compressed_bytes
value_updates	Counter	Number of updates to a topic that provide a full value.	diffusion_topics_value_updates
delta_updates	Counter	Number of updates to a topic that provide a partial value.	diffusion_topics_delta_updates
value_bytes	Counter	On each change of topic value, this metric increases by the size of the new value.	diffusion_topics_value_bytes

Metric name	Type	Description	Prometheus export
delta_bytes	Counter	On each change of topic value, this metric increases by the size of an internal delta representing the difference the previous and new values.	diffusion_topics_delta_bytes

Delta compression ratio

`value_bytes` and `delta_bytes` can be used to capture the theoretical delta compression ratio of the application data flowing through the topics. Both the console and the JMX MBean perform this calculation. The ratio is a value between 0 and 1. The closer the ratio is to 1, the more benefit the application data will obtain from delta streaming. If `value_bytes` is 0, there have been no updates, so the delta compression ratio is reported as zero. Otherwise it is calculated as:

```
1 - delta_bytes / value_bytes
```

Delta streaming is enabled for subscriptions by default, but can be disabled on a per-topic basis using the PUBLISH_VALUES_ONLY topic property. If delta streaming is enabled, a stable set of subscribers remain connected, and no session has a significant backlog (so conflation is not applied), the following relationship should hold:

```
subscriber_update_bytes # delta_updates x subscribers
```

Delta streaming can also be used to update topic values. If the delta compression ratio is high, but `delta_updates` is zero (or low, relative to `value_updates`), consider whether your application can use the stateful update stream API to take advantage of delta streaming.

Log metrics

Log metrics record information about server log events. Separate metrics are kept for each unique pair of log code and log severity level that has been logged.

The log severity levels are: error, warn, info, debug, trace.

A JMX MBean is created for each pair of log code and log severity that has been logged at least once.

Here is an example MBean name:
`com.pushtechnology.diffusion:type=LogMetrics,server="server_name",level=warncode=`

Session metrics versus network metrics

The network `inbound_bytes` and `outbound_bytes` metrics include bytes that are not counted by the equivalent session metrics.

The session metrics include bytes from transport framing and all session traffic (including additional HTTP traffic from long polling).

The network metrics include all bytes included in the session metrics as well as non-session bytes such as:

- TLS overhead
- Web server traffic (for example, browsers downloading the web console pages)
- Rejected connection attempts

Metrics in the Publisher API

Publisher metrics, client instance metrics, and topic instance metrics have all been removed.

Consequently the `PublisherStatistics`, `ClientStatistics` and `TopicStatistics` interfaces provide no information. These interfaces are deprecated and will be removed in a future release.

Limited server metrics are still available through the Publisher API using the `ServerStatistics` interface.

For more information, see the Java API documentation.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Diffusion management console on page 538
A web console for managing the Diffusion server.

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Integration with Splunk on page 615
How to achieve basic integration between Diffusion and the Splunk™ analysis and monitoring application

Configuring metrics

You can configure metric collectors using the console or JMX.

Metrics availability

Diffusion servers provide metrics which are made available in several ways:

- JMX MBeans
- through the Diffusion web console
- as endpoints for Prometheus

Metrics and performance

The cost of metrics has been greatly reduced compared to previous versions of Diffusion.

Topic and client instance statistics have been removed, and replaced with much more efficient metric collectors, which are suitable for production use.

Configuring metric collectors

Metric collectors allow custom aggregation of metrics that are relevant to your application. There are no default metric collectors, only the ones that you create.

There are two types of metric collector:

Session metric collectors

These can be configured to record metric data for a subset of all sessions, specified with a session filter.

The set of metrics recorded by each session metric collector is the same as those recorded for the whole server. For full details of session metrics, see the table in Metrics on page 531.

If the session filters of two different session metric collectors select the same session, both will record metrics for that session. It is only valid to add the metrics of different session metric collectors if their session filters select distinct sets of sessions.

You can optionally group the sessions within a collector by session properties.

Topic metric collectors

These can be configured to record metric data for a subset of all topics, specified with a topic selector.

You can optionally group the topics within a collector by topic type.

The set of metrics recorded by each topic metric collector is the same as those recorded for the whole server. For full details of topic metrics, see the table in Metrics on page 531.

If the topic selectors of two different topic metric collectors select the same topic, both will record metrics for that topic. It is only valid to add the metrics of different topic metric collectors if their topic selectors select distinct sets of topics.

You can create metric collectors using the Diffusion web console. See Diffusion management console on page 538 for further details.

You can also create metric collectors using JMX. The MetricCollectors on page 520 MBean has attributes which list the topic metric collectors and session metric collectors, together with operations that allow metric collectors to be created, replaced, and removed.

Note that metric collectors are replicated across a cluster unless `configurationReplication` is disabled in `Replication.xml`.

Metrics created by metric collectors

The JMX MBean object names used to publish metrics created by metric collectors have an additional property for the metric collector name, and further properties if the metric collector groups the results. Similarly, the Prometheus metric names used for metric collectors have additional dimensions for the metric collector names and grouping keys.

For example, a session metric collector with the name "My Session Metric Collector" that is not grouped by session properties, will publish its metrics to the JMX MBean:

```
com.pushtechnology.diffusion:type=SessionMetrics,server="server_name",
      name="My Session Metric Collector"
```

The Prometheus metrics have an additional `collector` dimension:

```
diffusion_sessions_connected{collector="My Session Metric Collector"}
```

```
diffusion_sessions_open{collector="My Session Metric Collector"}
```

, and so on.

Suppose the session metric collector is further grouped by the fixed session property `$Transport` and the application session property `Country`. Then there will be a separate MBean for each pair of property values. For example:

```
com.pushtechnology.diffusion:type=SessionMetrics,server="server_name",
      name="My Session Metric Collector",
```

```
      $Transport="WEBSOCKET","Country"="France"
```

The Prometheus metrics are similarly qualified, for example:

```
diffusion_sessions_connected{collector="My Session Metric Collector",
 _Transport="WEBSOCKET",Country="France"}
```

The dimension key in the Prometheus metric name has been adjusted to comply with Prometheus naming restrictions.

Similar naming patterns apply to topic metric collectors. For example, a topic metric collector with the name "My Topic Metric Collector" that is grouped by topic type will have a separate MBean for each matching topic type with a name like:

```
com.pushtechnology.diffusion:type=TopicMetrics,server="server_name",name="My
 Topic Metric Collector",Topic Type=JSON
```

The Prometheus metrics will be of the form:

```
diffusion_topics_subscriptions{collector="My Topic Metric
 Collector",type="JSON"}
```

The `Statistics.xml` configuration file

Metrics were previously configured in detail via `etc/Statistics.xml`. Changes to metrics in 6.3 mean that most of the configuration options in `Statistics.xml` are now deprecated.

- `<statistics>`

  The top-level element.
- `<client-statistics>`

  This section is used to configure the frequency of session reports to the server log.
- `<topic-statistics>`, `<publisher-statistics>` and `<server-statistics>` are now deprecated. Topic and server statistics have been replaced by metric collectors, which you can create using the console or MBeans.

In previous versions of Diffusion, the `<reporters>` element was used to configure how metrics were distributed.

These reporters have now been removed and the `<reporters>` element is deprecated. If any reporter configuration is found, a warning will be logged on start up.

It is not necessary to enable a JMX statistics reporter. JMX MBeans are now always enabled.

The topics and sessions reporters previously configured in `Statistics.xml` are now replaced with metric collectors. You do not need to enable metric collectors to create them.

Publisher API configuration

The use of the Publisher API is no longer supported.

Diffusion management console

A web console for managing the Diffusion server.

About

The Diffusion management console is an optional feature. It is deployed by default. It exists to give you an easy way to monitor and manage your Diffusion solution using a web browser.

To remove the console, you can delete the console files from `html/console` and `html/classic-console` within your installation directory.

Dependencies

The new console requires the latest version of a modern browser such as Chrome, Firefox, or Edge. Internet Explorer is no longer supported.

Classic console

Diffusion 6.5 includes a completely re-engineered console with an improved interface, better scalability and more help integration.

The "classic" console from previous versions is available at https://localhost:8080/classic-console. The classic console is no longer being actively developed and will be removed in a future release.

Logging in

The console is available in a fresh local installation at https://localhost:8080/console.

The console is secured by a principal (username) and password. The principal you use to log in must have permissions to view and act on information on the Diffusion server, for example by having the ADMINISTRATOR role.

The console can optionally be configured to allow users to log in with an external authentication provider. This is useful for integrating Diffusion with corporate single sign-on (SSO) systems. For more information, see Configuring the Diffusion management console on page 540.

The default configuration of the Diffusion server can be accessed with these credentials:

- principal: 'admin'
- password: 'password'

This user has the correct permissions to use all of the console's capabilities. For more information, see Pre-defined users on page 155.

> **Note:** We strongly recommend that you change the default security configuration before putting your solution into production. For more information, see Configuring security on page 446.

Features: Overview tab

The Overview tab of the console contains panels providing key information about the server.

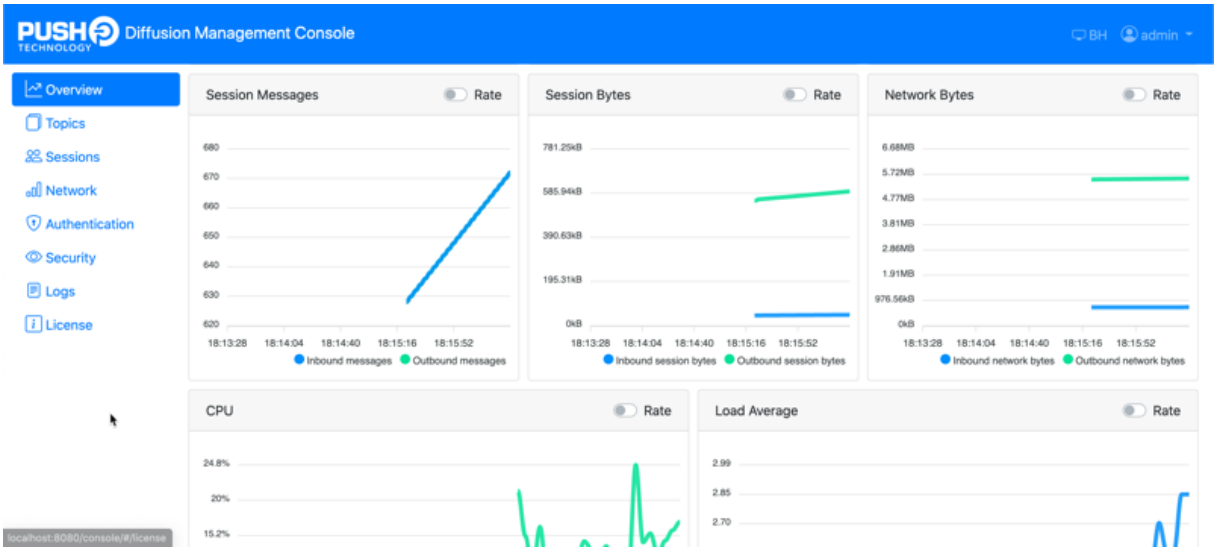


Figure 31: The new console Overview tab

Features: Topics tab

You can use this section to browse and interact with the Diffusion topic tree. You can browse the live topic tree, subscribe to topics and add/delete topics.

This tab also enables you to create topic metric collectors and topic views.

In the **Metric Collectors** section, specify the topics to include using the topic selector syntax. You can optionally choose to group by topic type. For more information about metric collectors, see Metrics on page 531 and Configuring metrics on page 535.

In the **Topic Views** section you can create a topic view using a topic view definition. See Topic views on page 50 for more information.

Features: Sessions tab

The Sessions tab shows a live list of the sessions connected to the Diffusion server in the **Open sessions** section, including session ID, IP address, connection and transport type, and total session time.

You can use the **Metric Collectors** section of this tab to configure a session metric collector. These enable you to gather information on a subset of all sessions. The **Metrics** section displays the output of your session metric collectors.

Each session metric collector provides information about the number of sessions (open, connected, peak and total), as well as inbound and outbound traffic in both bytes and number of messages. You can optionally group the sessions within a collector by session properties.

In the **Metric Collectors** section, specify the sessions to include using the session filter syntax.

You can group by session properties.

For more information about metric collectors, see Metrics on page 531 and Configuring metrics on page 535.

Features: Security tab

The Security tab shows a live list of security roles that are configured on the the Diffusion server system authentication handler. You can configure roles and isolated paths from this tab.

For more information about security, see Security on page 137.

Features: Logs tab

The Logs tab shows a live color-coded display of log entries emitted by the server at the levels of **INFO**, **WARN**, and **ERROR**.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Metrics on page 531
Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Integration with Splunk on page 615
How to achieve basic integration between Diffusion and the Splunk™ analysis and monitoring application

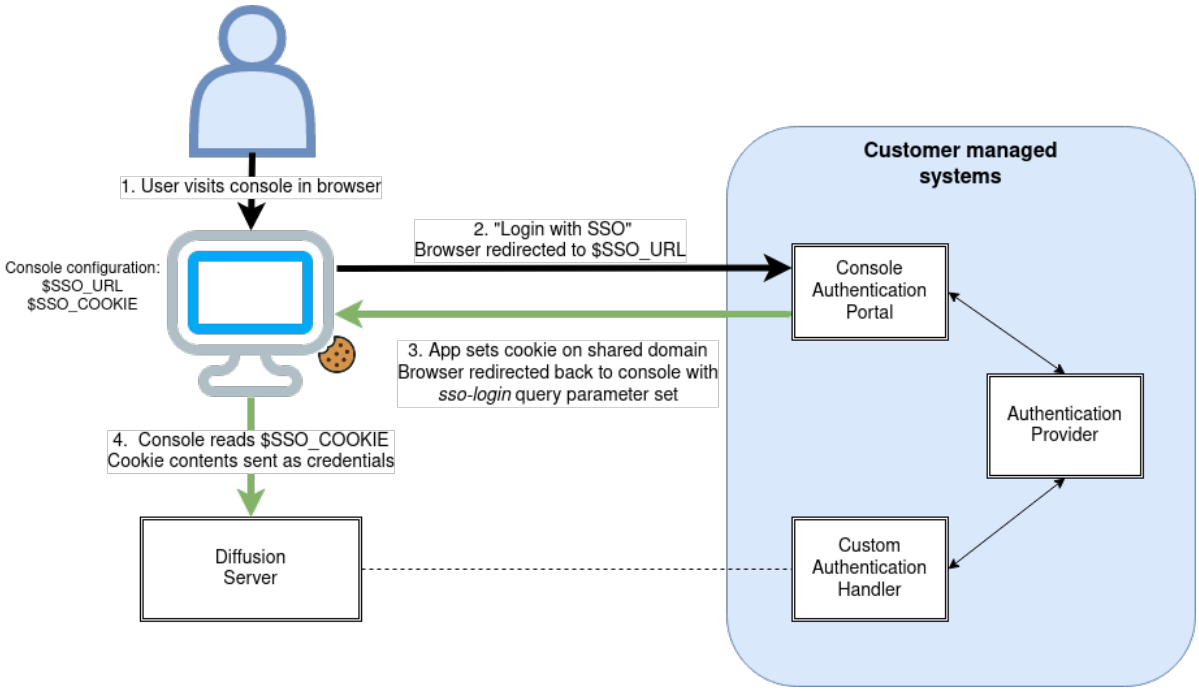
Configuring the Diffusion management console

The Diffusion management console can be configured by modifying the `html/console/config.js` file.

Single sign-on

The Diffusion management console can optionally be integrated with a single-sign on system. This is disabled by default.

Authentication is accomplished by obtaining an opaque token from a cookie that is passed to a custom authenticator. JWT tokens are useful in this scenario, but any binary data can be used.



Note: In order to this feature securely, the Diffusion server must be configured with an authentication handler which performs authentication with single-use tokens.

Writing console authentication portals

The console authentication portal is responsible for authenticating the user with an external authentication provider. This is usually performed with SAML, OAuth or OpenID connect.

If the user successfully authenticates, the console authentication portal must set a cookie on a domain such that it will be provided to the console by the user's browser. This may be accomplished by setting a cookie on a shared parent domain of the authentication portal and the console, or by using reverse proxies to present the authentication portal on the same domain as the console. The console authentication has total control of how the cookie is configured, but the name must match the value of the SSO_COOKIE field in `html/console/config.js`.

Once a cookie has been set, the console authentication portal must redirect the user's browser back to the Diffusion management console, with the query parameter *sso-login* present.

Note: In the interests of security, the following cookie attributes are recommended:

- **MaxAge** of 60 seconds, or as short as reasonably possible.
- The **Secure** attribute set, with the console available over HTTPS.
- Some browsers may log warnings if **SameSite** is not specified.

Writing custom authentication handlers for console authentication

A custom authentication handler must be configured in the Diffusion server. The console will send login requests to Diffusion with the principal **Diffusion Management Console SSO** and custom credentials containing the value stored in the cookie.

The authentication handler is then responsible for authenticating the session as usual, including setting the session principal and roles as appropriate. For more information, see Authenticating new sessions on page 342.

Logging

Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

> **Note:** The information in this section applies to logging that occurs at the Diffusion server. Some clients provide logging capabilities. For information about using logging with your Diffusion clients, see the Developer Guide section for the client API you are using.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Configuring log4j2 on page 455
To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

Configuring logging on the Diffusion server on page 454
Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

Configuring legacy logging on page 457
To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Related reference

Metrics on page 531
Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Diffusion management console on page 538
A web console for managing the Diffusion server.

Integration with Splunk on page 615
How to achieve basic integration between Diffusion and the Splunk™ analysis and monitoring application

Log messages on page 547
The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Log4j2.xml on page 456
Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

Logging using another SLF4J implementation on page 461

You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

Logging back-end

The work of formatting and writing out messages logged by the Diffusion server is done by the logging back-end. Diffusion uses Log4j2 as the default framework, but you can configure the Diffusion server to use other SLF4J implementations.

Log4j2 logging framework

Diffusion supports log4j2 as the default logging implementation. Log4j2 is a third-party SLF4J implementation provided by the Apache Software Foundation. For more information, see http://logging.apache.org/log4j/2.x/.

The log4j2 implementation of SLF4J supports a wide range of appenders and allows fine-grained tuning of logged events.

You can change the log4j2 logging behavior by editing the provided `log4j2.xml` configuration file.

For more information, see Configuring log4j2 on page 455.

Legacy logging framework

The legacy logging framework was the default in versions of Diffusion before 6.4. It is configured to log messages out to the console and write them to a file. You can configure the behavior of the legacy logging framework using the `Logs.xml` configuration file.

For more information, see Configuring legacy logging on page 457.

Other logging frameworks

Your Diffusion server can be configured to use any logging framework that implements SLF4J. However, only the log4j2 and legacy frameworks are supported for production use.

For more information, see Logging using another SLF4J implementation on page 461.

Related concepts

Configuring log4j2 on page 455
To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

Configuring logging on the Diffusion server on page 454
Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

Configuring legacy logging on page 457
To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Related reference

Log4j2.xml on page 456
Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

Logging using another SLF4J implementation on page 461

You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

Logging reference

Messages logged by the Diffusion server are logged at different levels depending on their severity.

Log levels

Diffusion events are logged at different levels of severity. The log levels, ordered from most severe to least severe, are as follows:

Table 48: Log levels

Level	Description
ERROR	Events that indicate a failure.
WARN	Events that indicate a problem with operation.
INFO	Significant events.
DEBUG	Verbose logging. Not usually enabled for production.
TRACE	High-volume logging of interest only to Push Technology Support. Push Technology Support may occasionally ask you to enable this log level to diagnose issues.

Warning: Logging can use considerable CPU resources. In a production environment, enable only significant log messages (`INFO` and above). Performance degrades significantly when running at finer logging levels as more messages are produced, each requiring processing.

Log format

Log messages output by Diffusion use the following default format.

Each log line is made up of a number of fields. All of the fields except for the Exception are formatted on a single line, delimited by pipe (`|`) characters.

```
yyyy-MM-dd HH:mm:ss.SSS|Level|Thread|Code|Principal|Session ID|
Message|LoggerName Exception
```

If you are using log4j2 as your logging back-end, you can edit the provided `Log4j2.xml` configuration file to change the log format. For more information, see Configuring log4j2 on page 455.

Note: Sometimes log messages that are output to the same location as Diffusion messages can be from other products. You can see which messages are Diffusion messages by looking for the message code of the format PUSH-*XXXXXX*. All messages that Diffusion outputs at INFO level or above include this code.

The meaning of each field is described in the following table.

Table 49: Fields included in the logs

Field	Optional or Mandatory	Format/values stable between releases	Description
Time stamp	Mandatory	Yes	The time and date the log event occurred. Asynchronous logging is enabled by default. The server might log a message in a different thread to the one that produced the log event, and at a slightly later time. Consequently, log lines might not be logged in exact time stamp order. The time stamp is displayed using the timezone configured for the JVM running the server. The date format can be changed in the `Server.xml` configuration file.
Level	Mandatory	Yes	The log severity, using the SLF4J levels: `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`.
Thread	Mandatory	No	The name of the Java thread that logged the event.
Code	Optional	Yes	Diffusion log messages have a unique code. For example, `PUSH-000123`. For more information, see Log messages on page 547. All messages at that are logged at INFO or above are documented.
Principal	Optional	Yes	The security principal used to authenticate the client session which originated the event (if any).

Field	Optional or Mandatory	Format/values stable between releases	Description
Session ID	Optional	Yes	The session ID (if the event was initiated by a client session).
Message	Mandatory	No	A natural language description of the event.
Logger name	Mandatory	No	The logger name. Usually the fully qualified name of the Java class that produced the event.
Exception	Optional	No	If the log event has an associated Java `Throwable`, the exception message and stack trace directly follows the message line.

Optional fields are empty if the log event does not have the information.

The third column indicates whether fields are stable between releases. Where possible, Push Technology will not change the format or values of these fields so they can be relied on for automated log monitoring. The fields not marked as stable are more likely to change between releases, including patch releases.

Log message examples

The following examples show the log format output by default, using log4j2 and the provided `Log4j2.xml` configuration file.

Most log messages are formatted on a single line.

```
2020-06-15 14:01:31.199|INFO|main|PUSH-000159|<control>|
7854be8b32b73230-0000000000000000|The maximum message size is 32768
 bytes.|com.pushtechnology.diffusion.DiffusionController
```

If a log event has an exception, the exception message and stack trace directly follows the message line. The exception can span multiple lines.

Notice that optional fields are left empty if they do not apply to a particular event.

```
2020-06-15 14:14:54.095|ERROR|main|PUSH-000164|||Diffusion Server not
 started.|com.pushtechnology.diffusion.api.server.DiffusionServer
com.pushtechnology.diffusion.server.security.persistence.store.StoreException:
 Error parsing SystemAuthentication.store
 at
 com.pushtechnology.diffusion.server.security.persistence.store.systemauthentica
 at
 com.pushtechnology.diffusion.server.security.persistence.store.AbstractFileProv
 at
 com.pushtechnology.diffusion.server.security.persistence.store.AbstractStoreImp
 at
 com.pushtechnology.diffusion.server.security.authentication.systemhandler.Syste
```

```
at
com.pushtechnology.diffusion.server.security.persistence.store.systemauthentica
at
com.pushtechnology.diffusion.server.security.authentication.AuthenticationManag
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.ja
at java.lang.reflect.Method.invoke(Method.java:498)
```

Log stopped

When the Diffusion legacy logging back-end rotates the log files, it outputs the message `Log stopped` at the end of the log file before creating a new log file and continuing to log messages in that new file.

This does not apply to log files output by log4j2 or other third-party SLF4J implementations.

Related concepts

Configuring log4j2 on page 455
To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

Configuring logging on the Diffusion server on page 454
Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

Configuring legacy logging on page 457
To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Related reference

Log4j2.xml on page 456
Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

Logging using another SLF4J implementation on page 461
You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

Log messages

The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Related concepts

Configuring log4j2 on page 455
To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

Configuring logging on the Diffusion server on page 454
Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

Configuring legacy logging on page 457

To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Log4j2.xml on page 456
Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

Logging using another SLF4J implementation on page 461
You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

Log messages

The Diffusion server outputs log messages. Each log message contains an ID, a message, and a description.

Related concepts

Configuring log4j2 on page 455
To use log4j2, ensure that the log4j2 JAR is at `/slf4j-binding.jar`. log4j2 is already located here when you first install the Diffusion server. Use the `log4j2.xml` configuration file to configure the behavior of log4j2.

Configuring logging on the Diffusion server on page 454
Your Diffusion installation provides the log4j2 logging framework, and a legacy logging framework. Configure the Diffusion server to use your preferred framework.

Configuring legacy logging on page 457
To use the legacy logging framework, ensure that the legacy logging JAR is at `lib/slf4j-binding.jar`. Use the `Logs.xml` configuration file to configure the behavior of the Diffusion legacy logging.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Log4j2.xml on page 456
Use the `Log4j2.xml` configuration file to configure the behavior of the log4j2 logging framework.

Logging using another SLF4J implementation on page 461

You can use other implementations of SLF4J for your logging. However, this is not supported for production use.

PUSH-000005

Exception from JMS provider '{}'.

Description

The JMS adapter received notification of an exception from the JMS server. Typically, this happens when the connection between the adapter and the server has been terminated.

Related concepts

JMS adapter on page 639
The JMS adapter for Diffusion, enables Diffusion clients to transparently send data to and receive data from destinations (topics and queues) on a JMS server.

PUSH-000017

Unable to create a subscription to '{}', exception is {}.

Description

The JMS adapter failed to create a subscription to the JMS destination associated with the given topic name.

Related concepts

JMS adapter on page 639
The JMS adapter for Diffusion, enables Diffusion clients to transparently send data to and receive data from destinations (topics and queues) on a JMS server.

PUSH-000024

Mime extension '{}' was already mapped to '{}' : overwriting map with '{}'.

Description

A configured Mime extension was already mapped to another value but has now been remapped to a new value. This indicates duplicate mime extension values in the mimes configuration.

Related reference

Mime.xml on page 482
This file specifies the schema for the mime properties.

PUSH-000040

WhoIs connection failure limit exceeded - not resolving.

Description

Five attempts to connect to the WhoIs have failed. There will be no WhoIs service.

Related reference

Server.xml on page 423

This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000041

Failure to connect to WhoIs provider at {}:{} - will retry 5 times.

Description

Unable to connect to the WhoIs provider indicated. The connection will be attempted up to five times before giving up.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000046

Client {} closing - {}. {}.

Description

A client session was closed.

PUSH-000047

Client {} closing - {}. {}.

Description

The given client is closing due to an exception.

Related concepts

Common issues when using a load balancer on page 630
There are some configuration options on your load balancer that can cause problems or inefficient behavior in your Diffusion solution.

PUSH-000056

{} license hard limit ({}) breached, connection rejected{}.

Description

The maximum number of licensed connections has been exceeded.

Related concepts

License restrictions on page 408
The Diffusion license can include restrictions on how the Diffusion server is used.

Related tasks

Updating your license file on page 409

You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

PUSH-000065

Failure processing message from server.

Description

An error has occurred at the client end of a connection to a server while processing a message from the server.

PUSH-000072

Connector '{}' started, listening on {}.

Description

A network connector started and is listening at the given address.

PUSH-000074

Connector '{}' received an HTTP connection but does not support HTTP.

Description

Connector has received an HTTP connection attempt but does not support HTTP connections. The connector does not define a valid web server.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000075

Unable to start connector '{}'.

Description

An error occurred while trying to start a connector.

PUSH-000076

Unable to start connector '{}', permission denied.

Description

The operating system denied permission for a resource used by this connector. Often this relates to reserved ports (less than 1024) on Unix-based operating systems.

Related reference

Connectors.xml on page 438

This file specifies the schema for the connectors properties.

PUSH-000077

HTTP Method '{}' is not supported in request from '{}'. Request: '{}'.

Description

An HTTP client has requested a method that is not supported by Diffusion.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000080

Connector '{}' only supports SSL connections. Rejecting non-SSL connection from '{}'.

Description

An attempt has been made to make a non-secure connection to a connector that is configured to only accept secure connections.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000081

Connector '{}' rejected connection from '{}' due to SSL handshake failure: {}.

Description

A secure (SSL) connection was attempted but the SSL handshake failed. This is commonly due to the client or browser not trusting the server certificate.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000082

Connector '{}' does not support SSL. Rejecting SSL connection from '{}'.

Description

An attempt has been made to make a secure (SSL) connection via a connector that does not support SSL. The connector does not define a keystore.

PUSH-000085

Connector '{}' failed to allocate input buffer of configured size ({}); allocated {}.

Description

The receive buffer of the socket was assigned a different amount of memory than requested. This is configured by the input buffer size. The configured input buffer size is a hint to the operating system. Refer to your OS documentation for any socket buffer limits. This can have performance implications.

Additional information

When you change the `input-buffer-size` in the `Connectors.xml` configuration file, this configures the following buffers:

- A buffer in the client multiplexer, which will be of the configured size
- A socket buffer managed by the operating system

Depending on the operating system configuration, the operating system might not provide you with a socket buffer of the specified size and you might be allocated a smaller one.

To ensure that the socket buffer is set to the same size as the input buffer in the client multiplexer, change your OS socket configuration.

On Linux

To see the current maximum size of the input socket buffer, run the following command:

```
sysctl -a | grep rmem_max
```

To set the maximum size of the input socket buffer, run the following command:

```
sudo sysctl -w net.core.rmem_max=number_of_bytes
```

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000086

Connector '{}' failed to allocate output buffer of configured size ({}); allocated {}.

Description

The send buffer of the socket was assigned a different amount of memory than requested. This is configured by the output buffer size. The configured output buffer size is a hint to the operating system. Refer to your OS documentation for any socket buffer limits. This can have performance implications.

Additional information

When you change the `output-buffer-size` in the `Connectors.xml` configuration file, this configures the following buffers:

- A buffer in the client multiplexer, which will be of the configured size
- A socket buffer managed by the operating system

Depending on the operating system configuration, the operating system might not provide you with a socket buffer of the specified size and you might be allocated a smaller one.

To ensure that the socket buffer is set to the same size as the output buffer in the client multiplexer, change your OS socket configuration.

On Linux

To see the current maximum size of the output socket buffer, run the following command:

```
sysctl -a | grep wmem_max
```

To set the maximum size of the output socket buffer, run the following command:

```
sudo sysctl -w net.core.wmem_max=number_of_bytes
```

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000087

Connector '{}' received an unidentified connection request [{}] from {}.

Description

An unidentified connection attempt has been made via a connector. The hexadecimal representation of the initial bytes of the connection request are logged in order to aid in identifying the origin.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000144

Routing topic failed to subscribe client {} to routing topic '{}' with target topic '{}'.

Description

A routing topic failed to subscribe a client to a topic.

Related concepts

Routing topics on page 86
A special type of topic, which can map to a different real topic for every client that subscribes to it. In this way, different clients can see different values for what is effectively the same topic from the client point of view.

PUSH-000151

Unable to get JMX MBean attribute.

Description

It was not possible to retrieve an attribute from a JMX MBean.

Related concepts

JMX on page 511

You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

PUSH-000152

Unable to load shutdown hook class '{}': {}.

Description

An error occurred loading the given third party class.

PUSH-000153

Unable to load startup hook class '{}': {}.

Description

An error occurred loading the given third party class.

PUSH-000154

The JVM has been signalled to shut down.

Description

The JVM has been signalled to shut down, most likely by the operating system.

PUSH-000155

No connectors have been configured. Creating a default client connector listening on port {} and a default high volume connector listening on port {}.

Description

No connectors have been configured therefore a default client connector and a default high volume connector (suitable for fan-out) have been created, listening on the specified ports.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000158

No thread pools configured. Created new pool definition called '{}'.

Description

No thread pools were configured therefore a default one has been added.

Related reference

Server.xml on page 423

This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000159

The maximum message size is {} bytes.

Description

The maximum message size has been established.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000160

No inbound pool has been configured - using '{}'.

Description

No inbound thread pool has been configured therefore the first configured thread pool has been assumed to define the inbound pool.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000162

Running shutdown hook '{}'.

Description

The given third party class is being executed at server shutdown.

PUSH-000163

Running startup hook '{}'.

Description

The given third party class is being executed at server startup.

PUSH-000164

Diffusion server not started.

Description

Diffusion server failed to start.

Related concepts

Starting the Diffusion server on page 484

After you have installed and configured your Diffusion server, you can start it using one of a number of methods.

PUSH-000165

Diffusion server started.

Description

Diffusion started successfully.

Related concepts

Starting the Diffusion server on page 484
After you have installed and configured your Diffusion server, you can start it using one of a number of methods.

PUSH-000166

Diffusion server '{}' starting.

Description

The Diffusion server is starting.

Related concepts

Starting the Diffusion server on page 484
After you have installed and configured your Diffusion server, you can start it using one of a number of methods.

PUSH-000167

Diffusion stopped.

Description

The Diffusion server has been stopped.

PUSH-000168

Diffusion stopping, reason='{}' by administrator='{}'.

Description

Diffusion is processing a shutdown request.

PUSH-000169

Diffusion - stopping connectors.

Description

Connectors are being stopped during a shutdown of Diffusion.

PUSH-000173

Exception notifying '{}' of '{}'.

Description

An exception occurred while processing an internal asynchronous event.

PUSH-000174

Unable to submit event '{}' to '{}' for execution.

Description

A failure has occurred while submitting a notification event for execution.

PUSH-000183

Message channel '{}' closed - {}.

Description

A communication error has occurred on a message channel.

PUSH-000185

Failed to accept connection on connector '{}'.

Description

This can occur when a socket connection has been made to Diffusion, but a failure occurred while initializing it.

PUSH-000188

Shutting down selector '{}' due to fatal error.

Description

An error has occurred in a selector thread and the selector will be shutdown.

PUSH-000191

Connector '{}' is unable to accept connection: {}.

Description

An exception occurred while attempting to accept a socket connection.

PUSH-000193

Connector '{}' has loaded a TLS keystore from '{}'.

Description

Connector has loaded TLS keystore from the specified location.

Related concepts

Network security on page 487

This section describes how to deploy network security, which can be used in conjunction with data security.

PUSH-000195

JMX: Cannot register object '{}' at MBean ObjectName '{}'.

Description

An error occurred registering the given object with the given JMX ObjectName with the JMX server.

PUSH-000199

License is not valid for this version of Diffusion (license='{}' vs '{}').

Description

The license file does not match this version of Diffusion.

Related concepts

License restrictions on page 408
The Diffusion license can include restrictions on how the Diffusion server is used.

Related tasks

Updating your license file on page 409
You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

PUSH-000201

{} license soft limit ({}) exceeded. Hard limit at {} connections{}.

Description

Once the number of connections for a product reaches its soft limit, a warning is emitted. The product might cease to function if this number reaches the hard limit.

Related concepts

License restrictions on page 408
The Diffusion license can include restrictions on how the Diffusion server is used.

Related tasks

Updating your license file on page 409
You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

PUSH-000202

Product license expires in {} day(s).

Description

The installed license file is nearing expiration. A new license will be required soon for Diffusion to continue running.

Related concepts

License restrictions on page 408
The Diffusion license can include restrictions on how the Diffusion server is used.

Related tasks

Updating your license file on page 409
You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

PUSH-000203

License has expired.

Description

The license has expired.

Related concepts

License restrictions on page 408
The Diffusion license can include restrictions on how the Diffusion server is used.

Related tasks

Updating your license file on page 409
You can update your Diffusion license file without having to restart the Diffusion server. Copy the new file over the old and ensure that the timestamp is updated.

PUSH-000206

Licensor stopping Diffusion. Invalid license.

Description

The Diffusion license is invalid.

PUSH-000212

Log level set to '{}' for '{}'.

Description

The logging level of the specified log file has been changed as indicated.

Related reference

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

PUSH-000214

Invalid JMX credentials.

Description

The supplied JMX credentials are incorrect.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

PUSH-000215

Remote JMX management service is disabled.

Description

The remote JMX service is configured not to start.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

PUSH-000216

Remote JMX management service has started. Listening to {}.

Description

The remote JMX service has started.

Related concepts

JMX on page 511
You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

PUSH-000226

Multiplexer error while processing client {}.

Description

A multiplexer tried to send messages for a client, but failed.

PUSH-000227

Sending event to multiplexer '{}' delayed as the event queue size is full.

Description

This thread is temporarily blocked by a multiplexer with a large backlog of events to process.

Additional information

The depth of the multiplexer event queue has exceeded the value of the `max-event-queue-size` element in the `Server.xml` configuration file.

PUSH-000228

Sending event to multiplexer '{}' is significantly delayed as the event queue is full.

Description

This thread has been blocked for a long time by a multiplexer with a large backlog of events to process. If this occurs frequently, capture a Java Flight Recording or thread dumps of the server and report to Push Technology.

PUSH-000229

Error while handling a multiplexer event.

Description

Error while handling a multiplexer event.

PUSH-000230

Failed to schedule event.

Description

A multiplexer was unable to schedule an event in another thread.

PUSH-000231

Long multiplexer cycle [cycle {}: processed {} events in {} ms; processed {} network operations in {} ms].

Description

A multiplexer processing cycle exceeded the configured notification threshold. Common causes include concurrent garbage collections (enable JVM garbage collection logging to investigate); overly general topic selectors that must be tested against many topics (prefer topic selectors with more specific prefix paths); subscription processing for many sessions (if more CPU cores are available, consider increasing the number of multiplexers); more multiplexers configured than available CPU cores (reduce the number of multiplexers). If the reported operation count is low, reducing the multiplexer monitoring period threshold will provide further diagnostics.

PUSH-000233

Multiplexer '{}' started.

Description

A multiplexer has started.

PUSH-000298

No queue definitions configured : Adding {}.

Description

No queue definitions have been configured therefore a default queue configuration is being assumed.

PUSH-000301

Default queue definition {} does not exist. '{}' used.

Description

There is no configured queue definition with the name as specified as the default queue therefore the first configured queue is assumed to be the default.

PUSH-000302

Default queue definition not configured. '{}' used.

Description

No default queue definition has been specified in the configuration therefore the first configured queue definition has been assumed to be the default.

PUSH-000303

Queue definition '{}' not known - using default.

Description

The specified queue definition is not known within the configuration therefore the default queue definition has been assumed.

PUSH-000310

Method '{}' from Diffusion servlet called, not being processed.

Description

The Diffusion servlet's implementation of the given method does nothing.

PUSH-000311

Diffusion servlet: {}.

Description

Indicates the startup status of the Diffusion servlet.

PUSH-000316

Failed to update statistics file '{}'.

Description

It was not possible to write the given statistics file. See the logged exception for details. The file directory is determined by the default-log-directory element of the etc/Logs.xml configuration file.

Related reference

Metrics on page 531
Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Statistics.xml on page 482

This file specifies the schema for the properties defining statistics collection.

PUSH-000335

Uncaught exception in thread '{}'.

Description

An uncaught exception has been thrown from a specified thread and logged.

PUSH-000337

Added file service virtual host '{}' to serve files from '{}' to requests where the host header matches '{}'.

Description

A virtual host has been added to the web server's file service.

PUSH-000338

Unable to process alias entry in file '{}': {}.

Description

Diffusion tried to add a new alias to the web server, but failed.

PUSH-000351

{}: Connection rejected as request was missing the HTTP header field '{}'.

Description

This connection was rejected as it was missing the given HTTP header. The connection is likely not from a Diffusion client, or an intermediary (firewall/load-balancer) has removed the header.

PUSH-000352

{}: Connection rejected as [{}] did not match '{}'.

Description

The connection from an untrusted host was rejected, in accordance with CORS configuration.

PUSH-000353

{}: Connection rejected as [{}] did not match '{}'.

Description

The connection was rejected because its WebSocket origin did not match the regular expression stored in the configuration.

PUSH-000354

{}: cors-origin not defined or invalid. Aborting request.

Description

CORS (Cross Origin Resource Sharing) request has been received by the web server client service but no "cors-origin" has been configured or the configured value was invalid. The request has been aborted.

PUSH-000355

{} unable to process HTTP Request '{}'.

Description

Web server file service was unable to process the given HTTP request.

PUSH-000356

Client service {}: HTTP processing error on connector '{}'.

Description

An error occurred handling an HTTP request.

PUSH-000357

{}: Invalid regular expression '{}' for CORS origin '{}' - feature disabled.

Description

An invalid regular expression was given for the CORS origin configuration.

Related concepts

Cross domain policies on page 625
Cross domain policies grant permission to communicate with servers other than the one the client is hosted on.

Related reference

WebServer.xml on page 474
This file specifies the schema for the web server properties.

PUSH-000359

Poll request for client {} from '{}' failed because no matching client session can be found. The client session might have been closed previously. Poll request will be ignored.

Description

The server has received a poll request for an unknown client session. This might be because the client has been closed by the server, or indicate an incorrectly configured load balancer routing an HTTP connection to a server that was not hosting the session. Consider enabling session-stickiness for HTTP clients.

PUSH-000360

Not adding file service virtual host '{}' to serve files from '{}' to requests where the host header matches '{}' as it is not a directory.

Description

Failed to add a virtual host to the web server's file service because the configured mapping does not point to a directory.

PUSH-000361

Unable to service HTTP request for service '{}'.

Description

An HTTP service was invoked, but it failed to process the incoming request.

PUSH-000362

Started HTTP service for '{}'.

Description

An HTTP service was successfully started.

PUSH-000364

Web server {} failed to instantiate HTTP service '{}' : {}.

Description

A web server failed to instantiate the named configured HTTP service.

PUSH-000366

Virtual host HTTP request {} is invalid.

Description

An HTTP request being handled by a virtual host is trying to break out of the virtual root.

PUSH-000367

Virtual Host HTTP request {} - unable to read file {}.

Description

A failure has occurred trying to read the specified HTML file.

PUSH-000370

WhoIs Provider class instantiation failure calling '{}'.

Description

Unable to instantiate the configured WhoIs provider. Will use the default WhoIs provider.

PUSH-000372

Unable to load GeoIP Database '{}'.

Description

An error occurred while loading or initializing the GeoIP database.

Related reference

Server.xml on page 423

This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000373

WhoIs Service failed to lookup address '{}'.

Description

A failure has occurred in the WhoIs service when trying to look up the given address.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000374

WhoIs service starting with {} thread(s).

Description

The WhoIs service is starting.

PUSH-000376

Deprecated element '{}' found in {}.xml.

Description

A deprecated element has been found in specified configuration file and should be removed.

PUSH-000380

Failed to parse {} at line {}, column {}: {}.

Description

An XML validation event has occurred while validating an XML property file.

PUSH-000381

Loaded XML {} properties from '{}'.

Description

The specified XML properties have been loaded from the file indicated.

PUSH-000386

Failed to load connection validation policy {} from {}.

Description

A failure has occurred loading XML connection validation policy from the specified file.

PUSH-000396

Unable to perform substitution within property value '{}' due to syntax error.

Description

A syntax error has been detected while trying to perform environment variable substitution on a configuration property.

PUSH-000399

Directory '{}' does not exist or is not a directory. Using '{}' as a log directory.

Description

The directory of a log definition or the default log directory does not exist or it is not a directory. In this case the temporary file directory is used instead. This is specified by 'java.io.tmpdir'. If the intended directory does exist the problem might be that the relative path is incorrect because the working directory is not what you expect. The simplest way to resolve this is to use absolute file paths to reference log directories. This message might be shown when the log is created and when checking if it needs to be rotated.

PUSH-000406

Application handler threw exception. [cid={}].

Description

An application handler threw an exception when called.

PUSH-000412

Executor {}: task finished abnormally.

Description

A background task has thrown an exception. See the log for more information.

PUSH-000415

Received request from client {} to close client {}.

Description

The server has received a request from a client to close a client session.

PUSH-000416

Received request from client {} to close client {}. The client session does not exist.

Description

The server has received a request from a client to close an unknown client session.

PUSH-000420

One or more clients have registered to handle queue events, but none was available for a {} event for client {}.

Description

No client was available to handle the queue event, or one was chosen and failed while processing.

PUSH-000421

Received request from client {} to set conflation for client {} to '{}'.

Description

The server has received a request from a client to change the conflation setting for a client queue.

PUSH-000422

Received request from client {} to set conflation for client {} to '{}'. The client session does not exist.

Description

The server has received a request from a client to change the conflation setting for an unknown client session.

PUSH-000426

{}: command service call failed, reporting error to {}: {} {}.

Description

An internal service call has failed. An error will be reported to the client session listener.

Additional information

This log message can be generated by a variety of situations. The SDK you are using may provide more information about what happened as an ErrorReason code.

Note that this message does not necessarily indicate a problem. It can be generated if the end user caused a session to close (for example, by closing their browser or navigating to a different page).

PUSH-000430

There are currently no controllers for {}.

Description

A control binding has been established, but there are no available controllers.

PUSH-000431

Rejected attempt by session {} to bind a control handler: {}.

Description

This might indicate that another control group has established a binding, or that the client already has a binding.

PUSH-000432

Attempt by session {} to register or unregister from an unknown control point: {}.

Description

This is unexpected and possibly indicates the server and client versions are incompatible.

PUSH-000435

Task failure in selector thread '{}'.

Description

A task failed to be executed in a selector thread.

PUSH-000436

Unable to create file handler for logger '{}'. Log messages will still appear in the console.

Description

A file handler was unable to be created for the requested logger. Messages will continue to be logged to the console, but will not be recorded to their own file.

PUSH-000439

The authentication manager failed to start because an authentication handler of class {} could not be created.

Description

An instance of a configured authentication handler could not be created.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

PUSH-000440

Authentication handler {} threw an exception, denying authentication to {}.

Description

A configured authentication handler failed to process an authentication request.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

PUSH-000444

Unable to read aliases file '{}'.

Description

The configured aliases file could not be read.

Related reference

Aliases.xml on page 479

This file specifies the schema for the aliases properties used in a web server.

PUSH-000445

Discarding WhoIs request for {} due to backlog. There are currently {} pending requests.

Description

A WhoIs request has been discarded because there is currently a large number of pending requests. The WhoIsListener will be notified with basic locale information based on the caller's IP.

PUSH-000457

Service threads ({} multiplexer, {} selector, {} logging) exceed the number of available CPU cores ({}).

Description

Assigning more service threads than there are available CPU cores might lead to degraded performance. Service threads include multiplexer, selector and logging threads which require a dedicated CPU core for optimal performance.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000458

Application classpath entry '{}' in is not a directory. Check user libraries in the server configuration.

Description

The names in the user library configuration must identify directories.

Related reference

Server.xml on page 423
This file specifies the schema for the server properties, as well as multiplexers, security, conflation, client queues, and thread pools.

PUSH-000460

Class loader failed to enumerate files in directory '{}'.

Description

An error occurred reading the files within the given directory.

PUSH-000461

Unable to add file '{}' to the class loader.

Description

A server class loader failed to load the given file.

PUSH-000462

Multiplexer '{}' overflowed before start up - event discarded.

Description

A multiplexer queue overflowed before the multiplexer was started.

PUSH-000464

Update failed for {} topic '{}' [data type={}]: {}.

Description

An error was encountered while attempting to update a topic.

PUSH-000465

Connector '{}' has pings disabled. Unresponsive HTTP polling clients will not be detected.

Description

The server relies on pinging in order to detect unresponsive HTTP polling clients. Unresponsive clients might go unnoticed if the ping frequency is not set.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000467

Unable to establish a tunnel through the proxy at {}:{}.

Description

There was a problem in establishing a tunnel through the specified proxy. There might be an issue in reading from or writing to the proxy.

Related concepts

Connecting through an HTTP proxy on page 199
Clients can connect to the Diffusion server through an HTTP proxy by using the HTTP `CONNECT` verb to create the connection and tunneling any of the supported transports through that connection.

PUSH-000468

Topic stream '{}' threw exception.

Description

A topic stream callback raised an exception when called.

Related concepts

Using streams for subscription on page 227

Register a stream against a set of topics to access values published to those topics. For a registered stream to access the value of a topic, the topic type must match the stream and the client must be subscribed to the topic.

PUSH-000469

Failure to forward message from {} to {} for message path '{}'.

Description

The server failed to forward a message from a client to another client. The message might have been delivered. See the log for further detail.

PUSH-000470

The server failed to forward a message from {} to unknown session {} for message path '{}'.

Description

The server failed to forward a message as the target client session does not exist.

PUSH-000472

Service call to server from session {} failed: {}.

Description

The server rejected a service call due to an error. See the log message for further details.

PUSH-000473

Service call to client {} failed: {}.

Description

The client rejected a service call due to an error. See the log message for further details.

PUSH-000474

Failed to deliver missing topic notification for subscription or fetch to selector '{}' by session '{}'.

Description

A request made by the server to a missing topic handler resulted in an error.

PUSH-000475

The '{}' secret key encryption algorithm is not available.

Description

The specified password encryption algorithm is not available which means that a less secure default mode of password encryption will be used.

Related concepts

Network security on page 487

This section describes how to deploy network security, which can be used in conjunction with data security.

PUSH-000476

No authentication handlers are configured. All sessions will be anonymous.

Description

No authentication handlers are configured. All sessions will be anonymous.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

PUSH-000477

Configurations that specify only control authentication handlers are invalid.

Description

If a control authentication handler is configured, you must also configure a local authentication handler (such as the system authentication handler) to allow a client that registers a control authentication handler to connect.

Related concepts

User-written authentication handlers on page 153
You can implement authentication handlers that authenticate clients that connect to the Diffusion server or perform an action that requires authentication.

PUSH-000478

"{}" at line {} column {}.

Description

An error occurred while parsing security store language commands at the indicated line and column number.

PUSH-000479

Recovery of {} after a write error failed - store must be manually recovered from backup.

Description

Writing of a store file failed and automatic recovery from backup was attempted but also failed. In this event the store has been disabled and must be recovered manually from the backup file.

PUSH-000480

Recovery of "{}" required as backup "{}" exists.

Description

A store backup file has been found which suggests that recovery of the named store might be required. Inspect the store file and replace it with the backup if necessary, and then delete the backup.

PUSH-000481

Errors detected while parsing {} file "{}".

Description

Errors were detected while parsing the specified persistent store file - a detailed list of the errors will follow in the log.

PUSH-000482

Error reading persistent store file "{}".

Description

A read error has occurred on a security persistence file.

PUSH-000483

Failure writing to the {} file {}.

Description

An error has occurred while writing to the named persistence store. The store file could be corrupt and might need to be recovered from a backup.

PUSH-000490

Session error handler {} threw an exception.

Description

An application session error handler threw an exception when called.

PUSH-000491

Session listener {} threw an exception.

Description

An application session listener threw an exception when called.

PUSH-000493

WhoIs Service failed unexpectedly.

Description

The WhoIs Service failed.

PUSH-000495

Will {} for session {} found no topics below the branch.

Description

A client session will to remove a branch found no topics below the branch to remove.

PUSH-000498

Connector '{}' has received data on {} that has not come from a proxy.

Description

The connector has been configured to require all connections use a proxy protocol but a connection has been made that has not provided proxy information.

PUSH-000503

A blocking operation failed because the multiplexers failed to process it within {} milliseconds.

Description

This indicates that the server is severely overloaded or deadlocked.

PUSH-000504

No control handler found for routing topic '{}', subscription for client {} will be deferred until one is available.

Description

A routing handler must be registered for routing subscriptions to complete.

PUSH-000506

Flow control pressure {}%.

Description

Flow control back pressure has reached this value.

PUSH-000507

Flow control off.

Description

Back pressure has reduced, flow control is no longer being applied.

PUSH-000510

Unable to rotate metrics file '{}'.

Description

The internal FileMetricsWriter was unable to rotate out the specified file.

PUSH-000511

Unable to write metrics to file '{}'.

Description

The internal FileMetricsWriter was unable to write to the specified file.

PUSH-000512

Session property {} not allowed.

Description

An attempt has been made to define a session property with a name that is not allowed.

PUSH-000515

Cannot place JMS session to provider {}.

Description

An error occurred when trying to place a JMS session to the given provider.

PUSH-000516

Cannot publish message to topic {}.

Description

An error occurred while publishing a Diffusion topic update.

PUSH-000518

Connected to JMS provider '{}'.

Description

The JMS adapter has connected to the JMS provider with the given name.

PUSH-000520

Exception closing JMS Connection to {}.

Description

An error occurred while the JMS connection was being closed.

PUSH-000528

Connected to JMS provider {} v{}, JMS {}.

Description

Recording JMS provider and version, and JMS API version in use.

PUSH-000532

Failed to initialize Diffusion client.

Description

Fatal error initializing the Diffusion client.

PUSH-000533

Request to remote routing subscription handler for client {} / routing topic '{}' failed.

Description

A request to a remote routing subscription handler failed.

PUSH-000534

Routing subscription for client {} to routing topic '{}' failed because no topic exists with resolved topic path '{}'.

Description

A routing topic failed to subscribe a client to a topic because the handler returned an unknown topic path.

PUSH-000536

Fan-out connection '{}' to primary server at '{}' has failed ({}).

Description

A fan-out connection has failed to connect to the primary server.

PUSH-000537

Fan-out connection '{}' (session {}) to primary server at '{}' has been lost.

Description

A fan-out connection to a primary server has been lost.

PUSH-000538

Fan-out connection '{}' will attempt to connect to '{}' again every {} milliseconds.

Description

A fan-out connection to a primary server could not be established or has been lost and will now try to connect again at the specified interval.

PUSH-000539

Fan-out connection '{}' (session {}) - unexpected session error : '{}'.

Description

A fan-out connection has received an unexpected session error.

PUSH-000543

Fan-out started.

Description

Fan-out processing has been started.

PUSH-000544

Fan-out link '{}' failed to create replicated topic '{}'.

Description

A failure has occurred creating a replicated topic for a specified fan-out link.

PUSH-000546

Fan-out update of topic '{}' by '{}' has failed with '{}'.

Description

A failure has occurred whilst updating a fan-out secondary topic.

PUSH-000547

Failed to close selector '{}'.

Description

An error occurred closing a selector.

PUSH-000548

Failed to interrupt thread for selector '{}'.

Description

A selector thread could not be stopped cleanly.

PUSH-000549

Stopping selector thread '{}'.

Description

A selector thread is being stopped.

PUSH-000552

Remote JMX management service could not be started.

Description

The remote JMX management service was not started correctly. The rmiConnectorServer is not active.

PUSH-000553

Multiplexer blocked because it has a maximum-sized batch of {} notifications to deliver to the publisher(s).

Description

A multiplexer has encountered a severe backlog dispatching publisher notifications. The notifications will be delayed. This message indicates the publishers cannot keep up with the rate of notifications. The server may be overloaded or the publisher may be blocked.

PUSH-000554

Delayed dispatch of {} notifications because the notification queue is full.

Description

A publisher is failing to keep up with the rate of multiplexer notifications. The server may be overloaded or the publisher may be blocked.

PUSH-000555

Closing '{}' because the session failed to consume messages fast enough and a {} byte message breached its {} queue limit. Queue details: {}.

Description

The messages in the outbound queue for a session have reached the configured limit of messages. The session will be closed. This can indicate problems with the network performance or the receiving process. Consider increasing the message limit, limiting the rate of messages sent (for example, applying throttling or conflation), tuning network buffer sizes, or otherwise changing the client to process incoming messages at a higher rate.

PUSH-000557

{}: closed.

Description

A session properties dispatcher was closed.

PUSH-000558

{}: draining queued updates.

Description

A session properties dispatcher is starting. It is about to process a backlog of pending events.

PUSH-000559

{}: initialised.

Description

A session properties dispatcher has completed initialisation.

PUSH-000560

{}: initialising. Sending {} initial client notifications.

Description

A session properties dispatcher is starting.

PUSH-000564

Fan-out connection '{}' has established session '{}' with primary server at '{}'.

Description

A fan-out connection to the primary server has been established.

PUSH-000566

Daily statistics have been written in the ConnectionCount file.

Description

The server is shutting down - the daily statistics registered so far have been written in the ConnectionCount file.

PUSH-000567

Memory to calculate delta message exceeds limit of {} bytes - the full message will be used.

Description

A binary difference calculation failed because insufficient memory was available. This is a rare condition that only occurs when there are many differences between two huge messages.

PUSH-000571

Failed to start required connector, '{}'.

Description

A required connector, in Connectors.xml, has not been started.

PUSH-000572

Fan-out connection '{}' is removing topics '{}'.

Description

A fan-out connection is removing a set of replicated topics.

PUSH-000575

HTTP poll rejected - Invalid message channel '{}' cannot be cast to HTTPMessageChannel.

Description

HTTP poll attempted on transport that does not support polling.

PUSH-000576

A third-party SLF4J logger has been configured in place of the Log4j 2.

Description

The Diffusion classpath has been modified to use a third-party SLF4J logging library.

PUSH-000577

No server log has been configured.

Description

The log configuration does not specify a server log. Messages will only be logged to the console (stderr).

PUSH-000578

{} {} "{}" - Build #{}, {}, {} {} {}.

Description

Diffusion and Java product versions information.

PUSH-000581

Client service {}: Unrecognized HTTP request received on connector '{}' from address '{}'. Request: {}.

Description

The server failed to understand an HTTP request.

PUSH-000587

The value of {} ms configured for '{}' is excessive and has been limited to {} ms.

Description

The maximum value of the connect and write timeouts has been limited. A future version of Diffusion will enforce the new limits by failing to accept configurations with excessive values.

PUSH-000588

The connection activity monitor has detected that the connection '{}' has been idle and it has been closed, attempting to recover.

Description

Clients can use a connection activity monitor to listen for the system ping sent by the server. If the expected system pings are not received the connection will be closed and the client will enter a recovery state.

PUSH-000589

SSL channel {} was closed with data still pending.

Description

SSL connection was closed with data still pending.

PUSH-000590

Host JVM/JDK failed to provide an expected feature {}.

Description

The JVM/JDK failed to behave as expected at runtime. Please check the platform is supported.

PUSH-000592

Sending {} message(s) of {} bytes to {} was delayed by {} ms.

Description

The message took an unreasonably long time to be sent. This could be caused by network backpressure, flow-control or application delays.

PUSH-000596

Unable to look up session by token in the data grid.

Description

A protocol 5 or above reconnection was attempted and has failed. See the exception for more information on the cause of the failure.

PUSH-000597

Unable to recover the session {} from the data grid.

Description

During session fail over Diffusion updates the data grid and recovers information from it. This update or recovery did not succeed both the data grid and the session may have stale information. See the exception for more information on the cause of the failure.

PUSH-000598

Unable to remove sessions from the data grid.

Description

A server failed and the sessions did not fail over within the timeout. The sessions were not removed from the data grid. They will continue to take up memory in the data grid. See the exception for more information on the cause of the failure.

PUSH-000599

Unable to remove session {} from the data grid.

Description

The session has been closed but not removed from the data grid. It will continue to take up memory in the data grid. See the exception for more information on the cause of the failure.

PUSH-000600

Unable to store session {} in the data grid.

Description

An attempt to store a new session in the data grid failed. The session is known to a single server. See the exception for more information on the cause of the failure.

PUSH-000601

Failed to replicate change to session {}.

Description

An attempt to update the data grid with changes to a principal, properties, subscription level or roles of a session failed. The data grid may have stale information. See the exception for more information on the cause of the failure.

PUSH-000602

Unable to replicate change to session principal, properties or subscription level for session {}. The session could not be found.

Description

An attempt was made to update the data grid with changes to the principal, properties or subscription level of a session but the session was unknown.

PUSH-000603

Unable to replicate change to topic selections for session {}.

Description

An attempt to update the data grid with topic selections for a session failed. The data grid may have state information. See the exception for more information on the cause of the failure.

PUSH-000604

Suppressed {} further {} messages.

Description

Repeated log messages have been suppressed.

PUSH-000605

An error occurred while stopping the remote JMX management service.

Description

The remote JMX management service was not stopped correctly.

PUSH-000608

Validation failed for connection {}.

Description

An connector validator rejected a connection attempt.

PUSH-000609

Reconnection aborted because server did not receive {} messages from {}.

Description

Reconnection aborted because messages sent from a client session were not recoverable.

PUSH-000610

Reconnection aborted because {} messages sent by the server were not received by {}.

Description

Reconnection aborted because messages sent to a client session were not recoverable.

PUSH-000611

Unable to complete reconnection, recovery failed for unknown session {}.

Description

A session could not be reconnected because it was unknown.

PUSH-000613

{} reconnected, but messages may have been lost.

Description

A session has re-establish communication with a server, but messages may have been lost. This can happen when reconnecting to another server in an HA cluster.

PUSH-000617

Connection attempt from {} to {} (connector '{}') rejected because it did not complete within the configured timeout of {} ms.

Description

A network connection timed out due to inactivity on the channel.

Additional information

The Diffusion server gives a client connection a limited time to complete its handshake processing. If the network connection takes longer than the timeout, the connection is closed and the Diffusion server PUSH-000617 is logged.

This issue can be caused by the following circumstances:

- Diffusion is heavily loaded.

  To reduce the number of refused connections, you can increase the connection timeout value in one of the following ways:

  - Update the `connection-timeout` element in the `Server.xml` configuration file to increase the default timeout value for all connectors. In the default configuration, this value is set to 5s. If the value is not defined here, a value of 2s is used.
  - Update the `connection-timeout` element in the `Connectors.xml` configuration file to increase the timeout value for a specific connector. If a timeout value is not defined for a connector, the value set in the `Server.xml` configuration file is used instead.
- Connections are being made by an application that pings or investigates the Diffusion connector ports without using the Diffusion API.

  If this is the case, you can suppress the log messages about invalid connections by using the `ignore-errors-from` element in the `Connectors.xml` configuration file to specify the source IP address of the invalid connection.

Related reference

Connectors.xml on page 438
This file specifies the schema for the connectors properties.

PUSH-000618

Input queue for inbound thread '{}' of size {} overflowed due to large number of connections. Risk of deadlock.

Description

The queue of work for an inbound thread overflowed. To avoid the risk of deadlock, reconfigure the inbound pool to have a queue at least as large as the number of concurrent connections.

PUSH-000621

Client {} closing - {} - {}. {}.

Description

A client session was closed.

PUSH-000622

Unsupported service {} requested by peer {}.

Description

The peer has requested an unsupported internal service. This can be due to a version mismatch between client and server. Some services are not unsupported for deprecated network protocols.

PUSH-000623

Fan-out connection '{}' (session {}) lost - attempting reconnection.

Description

A fan-out connection has been lost and is now attempting to reconnect.

PUSH-000624

Connection from {} to {} (connector '{}') closed ({}).

Description

A network communication error occurred. The connection has been closed.

PUSH-000627

Failed to redeliver missing topic notification for subscription or fetch to selector '{}' by session '{}' after {} attempts.

Description

Redelivery of a missing topic notification has been canceled after retrying a number of times.

PUSH-000628

Handler {} callback method threw an exception.

Description

A handler callback raised an exception when called. If the handler was open it has been closed with the CALLBACK_EXCEPTION ErrorReason. See the log for more information.

PUSH-000631

Fan-out connection '{}' could not propagate missing topic notification for subscription or fetch to selector '{}' by session '{}' because there is no connection to primary server at '{}'.

Description

Fan-out propagation of a missing topic notification failed because the primary server is disconnected.

PUSH-000632

Fan-out connection '{}' failed to propagate missing topic notification for subscription or fetch to selector '{}' by session '{}' to primary server at '{}'.

Description

Fan-out propagation of a missing topic notification failed.

PUSH-000633

Failed to redeliver missing topic notification for subscription or fetch to selector '{}' by session '{}' because there is no longer a suitable registered handler.

Description

Redelivery of a missing topic notification has been canceled by the server because there is no longer a registered handler.

PUSH-000634

'{}': reconnect attempt failed: {}.

Description

An attempt to reconnect has failed. Further attempts to reconnect may be made depending on the reconnection strategy.

PUSH-000635

'{}': reconnection rejected by server: {}.

Description

The server has rejected the reconnection attempt. No further attempts to reconnect will be made - the session will be closed.

PUSH-000636

'{}': reconnection failed due to timeout.

Description

The timeout to successfully reconnect has been reached. The session will be closed.

PUSH-000638

The '{}' file cannot be found.

Description

The store file was not found. A default store has been created.

PUSH-000642

Allocating larger buffer to accommodate a message larger than the default configured input buffer size. The new buffer size is {}. Consider increasing the input buffer size for channel {}.

Description

Client has sent a message larger than the input buffer size. A new buffer has been allocated to contain the message. If this happens frequently consider increasing the input buffer of the connectors to reduce buffer allocations.

PUSH-000643

Request {} callback method threw an exception.

Description

A request callback raised an exception when called. See the log for more information.

PUSH-000644

Completed diagnostic report to {}.

Description

A multiplexer diagnostic report has completed.

PUSH-000645

Failed to write diagnostic report to {}.

Description

A multiplexer diagnostic report could not be produced.

PUSH-000646

Starting diagnostic report to {}.

Description

A multiplexer diagnostic report has started.

PUSH-000648

Cluster member {} is connected to this server.

Description

A Diffusion server is connected to this server as a cluster member. It is possible to route service requests to it.

PUSH-000649

Failed to connect to cluster member {}.

Description

This server failed to establish a connection to another Diffusion server. The connection will be retried.

PUSH-000650

Cluster member {} is disconnected from this server{}.

Description

A Diffusion server is disconnected from this server. It is not possible to route service requests to it. The connection will be recreated.

PUSH-000651

Cluster member {} joined the cluster.

Description

A Diffusion server has joined the cluster. It has been discovered but may not be connected.

PUSH-000652

Cluster member {} left the cluster.

Description

A Diffusion server has left the cluster.

PUSH-000655

{} reconnected, and messages were recovered successfully.

Description

A session has re-established communication with a server, and in-flight messages were recovered.

PUSH-000656

Unable to start connector '{}' on port {}, address already in use.

Description

The address requested by the connector is already in use by a different process. Often this means that you already have a Diffusion server running or you have multiple connectors trying to use the same port.

PUSH-000657

Session{} failed to register or unregister a control handler: {}.

Description

A session failed to register or unregister a control handler.

PUSH-000658

Fan-out connection '{}' {}.

Description

A fan-out connection has changed to the specified state.

PUSH-000659

Fan-out connection '{}' request to change from {} to {} but current state is {}.

Description

A state change has been requested for a fan-out connection which is inconsistent with its current state. This may not be an issue but is reported for diagnostic purposes.

PUSH-000660

Fan-out link '{}' failed to start.

Description

A fan-out link has failed to start.

PUSH-000661

Fan-out link '{}' {}.

Description

A fan-out link has changed to the specified state.

PUSH-000663

Fan-out link '{}' subscription has failed : {}.

Description

A fan-out link subscription has been terminated.

PUSH-000664

Unhandled network processing failure during read '{}:'.

Description

An unhandled exception occurred while attempting to process incoming data. This is a bug, please report it to Push Technology.

PUSH-000666

Failed to apply delta for topic cache entry: {}.

Description

A session failed to apply a received delta to the current value it has for the topic.

PUSH-000667

Failed to convert value to {} for topic '{}'.

Description

A session failed to convert a received value to the type expected by a value stream.

PUSH-000668

Delta received before value for topic cache entry: {}.

Description

A session received a delta for a topic for which it has no current value.

PUSH-000670

Received request from session {} to change session properties for session {}. The client session does not exist.

Description

The server has received a request from a session to change the session properties for an unknown session.

PUSH-000671

A message received from another server in the cluster [{}] could not be parsed: {}.

Description

A message used to communicate between two members of a cluster cannot be parsed. The session between the two members will be closed. Check the servers are using compatible product versions.

PUSH-000673

Fan-out link '{}' failed to create replicated topic '{}' because the topic already exists and can not be removed.

Description

A failure has occurred creating a replicated topic for a specified fan-out link because a topic exists at the same path that fan-out can not take control of. This may be because the existing topic is owned by a publisher.

PUSH-000674

HTTP send failed for session {} - the message channel was already closed.

Description

HTTP send was attempted on a message channel that has already been shutdown.

PUSH-000675

{} was lost: {}.

Description

A communication error has occurred on an outbound connection.

PUSH-000676

This system has {} available cores which exceeds the license limit of {} cores. The server thread pools have been constrained to degrade performance.

Description

The system has more CPU cores than are allowed by the installed license. The server will continue to run with degraded performance. Please contact Push Technology to upgrade your license.

PUSH-000677

A request of datatype '{}' received on path '{}' is incompatible with request handler/stream '{}'.

Description

A session has received a request which is incompatible with the request handler/stream registered on its path.

PUSH-000679

Persistence restore failed to restore topic '{}'.

Description

The persistence service failed to restore a topic. Restore will continue but the topic will not exist.

PUSH-000680

Persistence restore failed to apply record {} to topic '{}' : {}.

Description

The persistence service failed to apply an update to a topic. Restore will continue but the topic state will be unknown.

PUSH-000681

'{}': connection failed and reconnect was not enabled.

Description

A connection failed to be established and reconnection was not enabled. The session will be closed.

PUSH-000682

Session {} closed due to {} - This is due to the session failing to respond to a ping from the server.

Description

The frequency of server pings is dictated by the system-ping-frequency element for each connector in Connectors.xml. Consider configuring this value.

PUSH-000683

Received corrupt topic replication data from cluster peer {}.

Description

Topic replication has failed.

PUSH-000684

Filter callback '{}' threw an exception.

Description

A messaging filter callback raised an exception in your application code. This occurred when attempting to process a response.

PUSH-000685

Exception whilst handling the request.

Description

The application threw an exception whilst handling a request.

PUSH-000686

Failed to convert value to {} received on path '{}' for request handler/stream '{}'.

Description

A session failed to convert a received value to the type expected by a request handler / request stream.

PUSH-000689

Deprecated attribute '{}' found on element {} in {}.xml.

Description

A deprecated attribute has been found in specified configuration file and should be removed.

PUSH-000691

A partition of the cluster log could not be compacted.

Description

Topic replication failed. The log is compacted to reduce the memory consumption but compaction failed. See the log for further detail.

PUSH-000692

A partition of the cluster log could not be recovered. The compacted log contains errors.

Description

Topic replication failed. The compacted log could not be loaded into the compacter. This implies it contains invalid data.

PUSH-000693

Request to the Kubernetes API failed.

Description

Request to the Kubernetes API failed.

PUSH-000694

Finding Hazelcast endpoints from Kubernetes registry at {}.

Description

Finding Hazelcast endpoints from the Kubernetes registry.

PUSH-000695

Found {} Kubernetes endpoints running Hazelcast, {}.

Description

Found Kubernetes endpoints running Hazelcast.

PUSH-000696

Feature '{}' is not licensed.

Description

A product feature is not licensed for this environment.

PUSH-000697

Atomic move of persistence file {} to {} could not be performed. A non-atomic move will be attempted.

Description

The move of a persistence file as an atomic file system operation could not be performed. A non-atomic move will be attempted, but if that fails then it could leave files in an inconsistent state.

PUSH-000698

File persistence compaction has failed. Compaction disabled.

Description

The file persistence service failed to compact files. Compaction has been disabled. Persistence files will now grow indefinitely. You should stop the server and address the problem.

PUSH-000699

File persistence logging has failed. Topic persistence logging has been disabled. Recovery required.

Description

The file persistence service failed to log and has been disabled. No further changes to topics will be logged. If the server is restarted, it will attempt to restore topics up to the point the logging failed, however if this fails then the failed persistence files will be copied to a recovery directory. If you do not want this to happen, delete the log files manually before restarting.

PUSH-000700

The file persistence service has successfully completed restore of topics.

Description

The file persistence service has finished restoring topics from files.

PUSH-000701

The file persistence service has failed whilst restoring topics.

Description

The file persistence service has failed whilst restoring topics from files. The server will continue but some or all persisted topics may not exist.

PUSH-000702

The file persistence service is restoring topics from {}.

Description

The file persistence service is starting to restore topics from files in the specified store directory.

PUSH-000703

Failed to handle Prometheus instrumentation request.

Description

Failed to handle Prometheus instrumentation request.

PUSH-000709

The replication configuration does not specify a connector for connections between servers in the cluster. Using the connector '{}'.

Description

An arbitrary connector has been selected for internal connections between servers in the cluster. To prevent this message, add a connector element to Replication.xml.

PUSH-000710

Did not find any Kubernetes endpoints, will retry in {}ms.

Description

Did not find any Kubernetes endpoints running Hazelcast.

PUSH-000711

Failed to add a topic because the number of topics would exceed the license limit of {}.

Description

The number of topics has reached the limit specified in the license.

PUSH-000712

Connector '{}' has scheduled pings every {}ms.

Description

The server relies on pinging in order to detect unresponsive HTTP polling clients.

PUSH-000714

There was a problem with the cluster log and Diffusion failed to write out records to the dump file {}.

Description

Topic replication failed. The cluster log contained invalid data and could not be written to the disk for debugging. This may happen if there are issues with multiple partitions, each partition races to stop the server and may interrupt other threads still writing out the file. It may also happen for other, file system related, reasons such as running out of disk space.

PUSH-000715

A request to {} failed. See the log for more information.

Description

A service request between members of the cluster failed. The logged exception should contain more information about the failure.

PUSH-000716

Fan-out connection '{}' session '{}' changed from '{}' to '{}'.

Description

The session state of a fan-out client connection has changed.

PUSH-000717

Failed to delete persistence temporary directory {}.

Description

Persistence service temporary files could not be deleted on server closedown. Large files might remain in the 'tmp' directory within the persistence directory. You can delete the files manually to recover file system space. They will be deleted automatically when the server is restarted with persistence enabled.

PUSH-000718

The server is not licensed to accept connections from remote servers.

Description

The license does not allow remote server or fan-out connections.

PUSH-000719

{} could not be logged to cluster partition {}.

Description

Writing to a cluster partition log failed. Failures known to be due to cluster repartitioning are logged at INFO level. Messages logged at WARN level merit further investigation.

PUSH-000720

Unsupported Log4j configuration. Logging may not be cleanly shut down on exit.

Description

Diffusion expects the configured Log4j library to be at least version 2.6. Earlier versions do not provide a public LogManager.shutdown() API.

PUSH-000732

Failed to update the license cluster pool statistics.

Description

The operation to update the cluster license pool statistics was completed with an error.

PUSH-000733

The file persistence service has failed to secure invalid files - manual recovery required.

Description

After failing to compact old files the persistence service moves such old files into a recovery directory. This file move has failed meaning invalid files may still exist in the persistence directory and cause future compaction or restore to fail. The server should be closed and the problem addressed by moving all files out of the persistence directory.

PUSH-000734

The file persistence service has failed to restore from existing files and has moved all old files to {}. These files will need to be manually deleted.

Description

The file persistence service failed to restore from old files and has moved the files into a recovery directory. The server will continue without having restored from any persistence files. The problem will need to be addressed and files should be manually deleted from the recovery directory.

PUSH-000740

{}.

Description

Regular log of client connection statistics.

PUSH-000743

Unable to bind outbound connection to local address {}.

Description

The application has requested a socket be bound to a specific local address but this is unsupported by the runtime platform. This will happen on Android versions earlier than API Level 24 ("Nougat").

PUSH-000744

The selector thread pool {} specified for connector {} was not found. Using default selector thread pool.

Description

A selector thread pool has been configured for a connector that has not been defined for the server. The default selector thread pool will be used.

PUSH-000747

Replacing loaded license [{}] with new license from '{}'.

Description

A new license file has been found and will be installed.

PUSH-000749

The server is waiting for the cluster to reach a quorum of {} servers.

Description

The start of the server has blocked while it waits for the quorum of servers to join the Hazelcast cluster.

PUSH-000750

The cluster is small enough for the quorum to mitigate split-brain. The cluster can't be divided into smaller clusters where more than one cluster can satisfy the quorum.

Description

The cluster is small enough to be protected against split-brain by the quorum. Split-brains may occur but they are mitigated by shutting down minority clusters.

PUSH-000751

The cluster is too large to completely mitigate split-brain. It is possible to divide the cluster into multiple clusters that satisfy the quorum requirements. The cluster has {} servers, should be less than {}.

Description

The cluster is too large to completely mitigate split-brain. It is possible to divide the cluster into multiple clusters that satisfy the quorum requirements. The cluster size should be smaller than twice the quorum size.

PUSH-000752

The cluster this server is in has lost enough members to lose the quorum. This server will shut down.

Description

The cluster this server is in has lost enough members to lose the quorum. The server previously had a quorum. It will now shut down. Other servers may be in a cluster with enough servers to satisfy the quorum.

PUSH-000753

The cluster has enough members to satisfy the quorum.

Description

The cluster has enough members to satisfy the quorum. It will now continue starting the server.

PUSH-000754

Failed to satisfy the quorum within the timeout. Not enough servers joined the cluster to satisfy the quorum. The server is shutting down without completing start up.

Description

Failed to satisfy the quorum within the timeout. Not enough servers joined the cluster to satisfy the quorum. The server has not completed starting up and the process will exit.

PUSH-000755

Record topic compatibility mode is enabled.

Description

Subscriptions to recordV2 topics by pre-6.0 clients will be presented as record topics.

PUSH-000756

Failed to add topic "{}" - REMOVAL property "{}" parse failure "{}".

Description

Adding a topic failed because parsing of the REMOVAL property failed.

PUSH-000758

Topic removal for topic '{}' failed with {} : {}.

Description

Removal of topics was attempted due to the removal policy of the named topic being satisfied but failed to complete.

PUSH-000759

Removing topics matching '{}' due to topic removal policy of topic '{}' created by '{}'.

Description

A removal policy specified on creation of the named topic by the named principal has been satisfied and the selection of topics covered by the given selector are now being removed.

PUSH-000760

Removing topic '{}' created by '{}' due to its topic removal policy.

Description

A removal policy specified on creation of the named topic by the named principal has been satisfied and the topic is now being removed.

PUSH-000761

Default web server '{}' created.

Description

A default web server has been created to support default connectors.

PUSH-000763

Received request from session {} to change session authorization roles for session {}. The client session does not exist.

Description

The server has received a request from a session to change the authorization roles for an unknown session.

PUSH-000765

Configured fan-out connection with url '{}' has no name. The name will default to the url.

Description

A configured fan-out connection has no name specified and so the name has defaulted to the url. A name will be mandatory in a future release and so should be specified.

PUSH-000766

Configured fan-out link for connection '{}' with selector '{}' has no name. The name will default to the selector string.

Description

A configured fan-out link has no name specified and so the name has defaulted to the selector string. A name will be mandatory in a future release and so should be specified.

PUSH-000767

This server is licensed to {}. License ID {}.

Description

The license used in this server is a commercial license.

PUSH-000768

The license file ({}) has been removed. You should replace it with a valid license, otherwise you will be unable to restart the server.

Description

The license file has been removed. You should replace the file with a valid license, otherwise you will be unable to restart the server.

PUSH-000769

The server has no IP address that satisfies the license constraint: {}.

Description

The license does not match the available network IP addresses.

PUSH-000770

The server has no MAC address that satisfies the license constraint: {}.

Description

The license does not match the available network MAC addresses.

PUSH-000771

The license file ({}) could not be loaded ({}). You should replace the file with a valid license, otherwise you will be unable to restart the server.

Description

The license file is invalid. You should replace the file with a valid license, otherwise you will be unable to restart the server.

PUSH-000772

License check failed: unable to resolve server network addresses.

Description

The server's network addresses could not be resolved.

PUSH-000773

Multiplexer {} is making slow progress on cycle {}. {} events processed, {} on queue. Recent history [time:+cycles,+idle cycles,+events,+network operations,queue]: {}.

Description

Diagnostic information logged when a multiplexer operational cycle has taken more than the configured monitoring period. Common causes include concurrent garbage collections (enable JVM garbage collection logging to investigate); overly general topic selectors that must be tested against many topics (prefer topic selectors with more specific prefix paths); subscription processing for many sessions (if more CPU cores are available, consider increasing the number of multiplexers).

PUSH-000774

Persistence file has a topic specification with legacy topic type STATELESS. Topics referencing this specification will be created as binary with DONT_RETAIN_VALUE set to 'true'.

Description

The persistence file being read has a record of a topic specification of type STATELESS. This topic type was removed in release 6.2. Persisted topics referencing this specification will be restored as binary topics that do not retain their value.

PUSH-000775

Authentication handler has specified an invalid $Roles property value : {}.

Description

An authentication handler has returned a value for the $Roles property that cannot be parsed. The roles for the session being authenticated will not have been changed.

PUSH-000776

The serialised object {} failed to restore.

Description

A failure occurred whilst restoring a serialised object. This can occur when recovering a partition or or when receiving an object from another peer in the cluster. Processing will continue but the named object may not exist or may not be consistent with other cluster members.

PUSH-000777

To prevent potential deadlock, a CompletableFuture has completed exceptionally because a get() or join() method was called from a Diffusion-managed thread.

Description

CompletableFutures returned by the API disallow calls to the blocking get() or join() methods from Diffusion threads to to prevent deadlocks.

PUSH-000778

{} failed to add the topic reference with path '{}'.

Description

A failure has occurred while adding a topic reference. See the log for more details.

PUSH-000779

Event diagnostic recording will now finish at {} ({} milliseconds from now).

Description

The multiplexer event diagnostic recording duration has changed.

PUSH-000780

Event diagnostic recording for preceding {} milliseconds. Event type,count,total time (ns),average time (ns) {}

Description

A multiplexer event diagnostic recording report has finished.

PUSH-000781

Recording event diagnostics until {} ({} milliseconds from now).

Description

A multiplexer event diagnostic recording has started.

PUSH-000782

Restore of {} from file failed.

Description

A failure occurred whilst restoring an object from file. Processing will continue but the object in question will not have been restored.

PUSH-000783

Store file '{}' needs recovery : backup file is '{}'.

Description

A persistence store file is in need of recovery due to a previous failure. The named file was backed up prior to the failure to the named backup file which can be copied back to the store file but items from the previous server may have been lost.

PUSH-000784

Terminal failure during the periodic compaction of persistence store '{}' : backup file saved in '{}'.

Description

An error has occurred whilst compacting a persistence store file leaving the store in an indeterminate state : A backup file has been created for recovery purposes.

PUSH-000785

Failure in periodic compaction of persistence store '{}'.

Description

An attempt to compact a persistence store file has failed - this will not affect the store but no more attempts at compaction will occur during the lifetime of the server.

PUSH-000786

Failure to read from persistent store file '{}'.

Description

Unable to read the specified store file. This may mean that the file is corrupt and the items within it may be lost. It must either be recovered from a backup or removed before the server can be restarted.

PUSH-000787

Failure to write to persistent store file '{}' and its state is unknown : Recovery file is '{}'.

Description

A write to the named store file has failed and logging to the file has been stopped. If the file existed previously then a backup will have been saved. The server should be closed and the reason for the failure diagnosed. The store file should be restored from the backup file before restarting the server.

PUSH-000790

Topic persistence is enabled.

Description

Topic persistence is enabled. Topics and their values will be persisted to file and automatically restored when server is restarted.

PUSH-000791

Session replication is enabled.

Description

Session replication is enabled. Connected session details will be replicated across the cluster.

PUSH-000792

Topic replication is enabled for all but : {}.

Description

Topic replication is enabled for all topics except those selected by the specified topic branches.

PUSH-000793

Topic replication is enabled for : {}.

Description

Topic replication is enabled for the specified topic branches.

PUSH-000794

Multiplexer {} is very busy. No idle cycles for {} periods of {} ms. Current [cycles,+idle cycles,+events,+network operations,queue]: {}. Recent history [time:+cycles,+idle cycles,+events,+network operations,queue]: {}.

Description

Diagnostic information logged when a multiplexer has had no recent idle cycles. If more CPU cores are available, consider increasing the number of multiplexers.

PUSH-000795

Configuration replication is enabled.

Description

Configuration replication is enabled. Items such as security stores, topic views and metric collectors will be distributed across the cluster.

PUSH-000796

Topic replication is enabled.

Description

Topic replication is enabled for all topics.

PUSH-000797

Failed to read attribute '{}' of MBean '{}'.

Description

The JMX service failed to read an MBean attribute.

PUSH-000798

Unexpected multiplexer recursion, threshold={}, capacity={}. Please report to Push Technology.

Description

An unexpected code path has been encountered. This is a bug, please report it to Push Technology.

PUSH-000801

Connector '{}' stopped.

Description

A network connector stopped.

PUSH-000802

Connector '{}' will be started when the following start conditions are satisfied: {}.

Description

A connector is waiting for its configured start conditions before starting.

PUSH-000803

Starting connector '{}' now start conditions are satisfied: {}.

Description

A connector is starting now its configured start conditions are satisfied.

PUSH-000804

The Diffusion log implementation is deprecated. Use Log4j 2 instead.

Description

The Diffusion log implementation is deprecated. The configuration should be updated to use Log4j 2.

PUSH-000805

The file service is serving files at http://{}:{} .

Description

The file service is available on a non-TLS enabled connector.

PUSH-000806

The file service is serving files at https://{}:{} .

Description

The file service is available on a TLS enabled connector.

PUSH-000807

{} delta quality from {} to {} (thread is {}% busy performing delta calculations).

Description

Reports an action taken by the server to balance CPU cost against binary delta quality.

PUSH-000808

JMX has loaded a TLS keystore from '{}'.

Description

The JMX connector has loaded a TLS keystore.

PUSH-000809

Remote server '{}' connected (session '{}') to server at '{}'.

Description

A remote server connection to the server at the specified URL has been established.

PUSH-000810

Remote server '{}' connection to server at '{}' has failed ({}).

Description

An attempt to establish a remote server connection to the specified server has failed.

PUSH-000811

Remote server connection '{}' (session {}) to server at '{}' has been lost.

Description

A remote server connection that was connected has been unexpectedly disconnected.

PUSH-000812

Remote server connection to '{}' {}.

Description

A remote server connection has changed to the specified state.

PUSH-000813

Remote server '{}' (session {}) lost - attempting reconnection.

Description

A remote server connection has been lost and is now attempting to automatically reconnect.

PUSH-000814

Remote server '{}' will attempt to connect to '{}' again every {} milliseconds.

Description

A remote server connection could not be established or has been lost and will now try to connect again at the specified interval.

PUSH-000815

Remote server '{}' (session {}) - unexpected session error : '{}'.

Description

A remote server connection has received an unexpected session error.

PUSH-000816

Remote server '{}' session '{}' changed from '{}' to '{}'.

Description

The session state of a remote server has changed.

PUSH-000817

Upgraded security store from language version {} to version {}.

Description

The server was started with a security store from a previous version of Diffusion, using an older version of the security language. The security store has been transformed to the current version of the security language.

PUSH-000818

Invalid JSON/CBOR value presented to topic '{}' - null value assumed.

Description

A value for a JSON topic that cannot be parsed as CBOR has been presented to a topic view evaluation. The evaluation will proceed as if the value was null.

PUSH-000819

IO error removing the delay queue file {}.

Description

An IO error occurred while removing a delay queue file. See the log for further details.

PUSH-000820

IO error reading from or writing to a topic view delay buffer [{}]. Delayed topic views will not function properly until the server is restarted.

Description

An IO error occurred. See the log for further details. Delayed topic views will not function properly until the server is restarted.

PUSH-000821

Invalid HTTP send received for session {} - '{}' is not a long polling message channel.

Description

HTTP long polling send received for a session that has an incompatible message channel.

PUSH-000822

Remote Server '{}' subscription has failed : {}.

Description

A remote server subscription has been terminated.

PUSH-000823

Remote server connection '{}' changed from '{}' to '{}'.

Description

The state of a connection from a remote or fan-out server has changed.

PUSH-000824

{} of {} could not be logged to cluster partition {}.

Description

Writing to a cluster partition log failed. Failures known to be due to cluster repartitioning are logged at INFO level. Messages logged at WARN level merit further investigation.

PUSH-000825

{} failed to queue delayed removal of {}. The topic reference will be removed immediately.

Description

A failure has occurred while removing a delayed topic reference. See the log for more details.

PUSH-000826

{} failed to queue delayed update of {}. The topic reference will be removed immediately.

Description

A failure has occurred while updating a delayed topic reference. See the log for more details.

PUSH-000827

Invalid value for topic {} dumped to file {}.

Description

An invalid value was passed to the named topic and dumped to the named file.

PUSH-000828

Invalid value for topic {} could not be dumped to file.

Description

An invalid value was passed to the named topic, but an attempt to dump the value to file failed.

PUSH-000829

Connector '{}' rejected connection from a {} because the server has not yet recovered shared configuration from the cluster.

Description

An attempt to establish a session was rejected because the server has not yet completed recovering shared configuration from the cluster. Consider configuration a cluster-configuration-restored start condition for the connector to avoid opening the network port until the server is ready.

PUSH-000830

An empty license file ({}) has been received. The license has not been changed.

Description

An empty license file has been received. You should replace the file with a valid license, otherwise you will be unable to restart the server.

PUSH-000831

{} failed to queue delayed addition of {}.

Description

A failure has occurred while adding a delayed topic reference. See the log for more details.

PUSH-000832

An unknown error occured when replacing the license file {}, with error {}.

Description

An unknown error occured when replacing the license file. The license has not been updated.

PUSH-000833

The license file ({}) could not be verified ({}). The license has not been updated.

Description

The license file could not be verified. The license has not been updated.

PUSH-000834

The server has restored shared configuration from the cluster.

Description

The server has joined a cluster and restored the cluster-wide configuration.

PUSH-000835

The server has restored replicated topics from the cluster.

Description

The server has completed recovery of topics from the cluster.

PUSH-000836

Remote server connection from server '{}' to itself rejected.

Description

A remote server connection from a server to itself has been rejected.

PUSH-000837

The security store file "{}" is read only.

Description

The security store file is read only, and is unable to be written to. Any changes will not be persisted and will be lost on restart.

PUSH-000838

The security store file "{}" is writable.

Description

The security store file is now writable. Any changes made to the security store will now be persisted.

PUSH-000839

The new license changes the enabled product features from {} to {}. Restart the server so the license takes effect.

Description

A new license has been installed that enables a different set of product features. Restart the server to ensure the license takes effect.

PUSH-000840

Replacing loaded license [{}].

Description

A new license has been uploaded to the server and will be installed.

PUSH-000841

Insert operation for topic view '{}' failed with invalid array index '{}'.

Description

A topic view insert operation has failed because the specified array index is beyond the bounds of the input array. The index must address an existing entry or be 1 greater than the last entry. To append to an array the '-' key value should be used.

PUSH-000842

Insert operation for topic view '{}' failed because parent element not found for '{}'.

Description

A topic view insert operation has failed because the parent of the specified insertion key could not be found. An insertion (as opposed to a replacement) can only be performed if the parent object or array exists in the input value.

PUSH-000843

Topic {}, state {} cannot have delta applied by {}.

Description

A delta could not be applied because the topic was in an incompatible state.

PUSH-000844

Topic {}, state {} cannot be updated by {}.

Description

An update could not be applied because the topic was in an incompatible state.

PUSH-000845

Persistence restore did not restore topic '{}' as it was an obsolete topic type.

Description

The persistence service did not restore the named topic as it was of an obsolete type.

PUSH-000846

Persistence file has a topic specification with legacy topic type SLAVE. Topics referencing this specification will not be restored.

Description

The persistence file being read has a record of a topic specification of type SLAVE. This topic type was removed in release 6.6. Persisted topics referencing this specification will not be restored.

PUSH-000847

The {} protocol is disabled for connector '{}'. Rejecting connection request from {}.

Description

A connection attempt was rejected because the protocol is disabled in the connector configuration.

PUSH-000848

Ignoring the 'No Local' option set in MQTT subscription from {}.

Description

The MQTT 'No Local' subscription option is not supported.

PUSH-000849

Ignoring the {} property set in MQTT {} from {}.

Description

An MQTT client sent a control packet with a property that the server does not process.

PUSH-000850

Invalid MQTT topic filter '{}' ({}) received from {}.

Description

An MQTT client has sent an invalid topic filter.

PUSH-000851

Rejecting MQTT connection request containing Will Message from {}.

Description

MQTT Will Messages are not supported.

PUSH-000852

Rejecting MQTT connection with unsupported protocol version {} from {}.

Description

An MQTT CONNECT packet was rejected because the MQTT version is not supported.

PUSH-000853

Failed to convert value of {} topic {} to UTF-8. MQTT subscribers will be sent a binary value instead.

Description

A topic's value could not be converted to a UTF-8 encoded string for MQTT sessions.

PUSH-000854

Closing MQTT session {} as session {} has been opened with the same client identifier ({}).

Description

An MQTT session has been closed because a new session has been opened with the same client identifier.

PUSH-000855

Unexpected failure while checking server {} for MQTT sessions with the same client identifier ({}).

Description

Failed to communicate with another server in the cluster while searching for MQTT sessions with the same MQTT client identifier as a new session.

PUSH-000856

MQTT session {} has the same client identifier ({}) as a newly opened session {} but won't be closed because it was authenticated with a different principal.

Description

MQTT session take over is restricted to sessions authenticated with the same principal.

PUSH-000857

The deprecated service {} was called by {}. The service was deprecated in version {}. Remove the use of deprecated APIs from client applications so they remain compatible with future server versions.

Description

A deprecated API has been used. Please review client applications for the use of deprecated APIs to ensure they remain compatible with future versions of the server. This warning is logged once for each deprecated service that is used.

PUSH-000858

The deprecated service {} was used to call {}. The service was deprecated in version {}. Remove the use of deprecated APIs from client applications so they remain compatible with future server versions.

Description

A deprecated API has been used. Please review client applications for the use of deprecated APIs to ensure they remain compatible with future versions of the server. This warning is logged once for each deprecated service that is used.

PUSH-000859

Failed to evaluate difference between values of length {} bytes and {} bytes with limit {}.

Description

A binary difference calculation failed unexpectedly. If this happens regularly, please report to Push Technology.

PUSH-000860

Failed to interrupt multiplexer {}.

Description

The server is shutting down and failed to gracefully stop a multiplexer thread.

PUSH-000861

Rejected unexpected connection attempt from {}. Check cluster members are configured with unique network addresses.

Description

A unexpected connection from a cluster peer was rejected. This can happen if two servers are configured with conflicting network addresses.

PUSH-000862

A missing topic notification for subscription or fetch to selector '{}' by session '{}' timed out.

Description

A request made by the server to a missing topic handler timed out.

PUSH-000863

Failed to convert MQTT payload received for {} topic {} due to data format error: {}.

Description

An MQTT session provided a payload that could not be converted to a Diffusion topic value.

PUSH-000864

Invalid property mappings in topic view '{}' prevents topic '{}' from being selected by view.

Description

One or more topic property mappings in a specified topic view contain invalid values and this has prevented the specified topic from being selected as a source topic for the view.

PUSH-000865

Remote server connection '{}' could not propagate missing topic notification for subscription to selector '{}' by session '{}' because there is no connection to the primary server.

Description

Remote server propagation of a missing topic notification failed because the primary server is disconnected.

PUSH-000866

Remote server connection '{}' failed to propagate missing topic notification for subscription to selector '{}' by session '{}'.

Description

Remote server propagation of a missing topic notification failed.

PUSH-000872

Unexpected multiplexer failure. {}.

Description

The server has corrupt multiplexer data structures. A diagnostic report has been created. This is a bug, please report to Push Technology.

Connection counts

The Diffusion server produces connection summaries.

At one minute past midnight Diffusion creates an entry in the file `logs/ConnectionCount`, and resets the counter.

The value in the third column is the number of new client connections that have been established that day.

The value in the fourth column is the maximum number of concurrent sessions that have been active that day.

> **Note:** The fourth column value (maximum number of concurrent sessions) can be higher than the third column value (number of new client connections) because of sessions established on previous days which are still active.

If you shut down the Diffusion server, the server updates this file with the client connection information for the day up to the point of shutdown. However, if the Diffusion server is killed instead of shut down, it does not update the file.

An example is shown here:

```
2022-09-27 00:01:40     6.7.7_01     128     31
2022-09-28 00:01:48     6.7.7_01     139     28
2022-09-29 00:01:56     6.7.7_01     177     28
2022-09-30 00:01:05     6.7.7_01     118     41
2022-10-01 00:01:22     6.7.7_01     207     36
2022-10-02 00:01:31     6.7.7_01     188     19
2022-10-03 00:01:41     6.7.7_01     244     44
2022-10-04 00:01:41     6.7.7_01     188     26
2022-10-05 00:01:41     6.7.7_01     195     39
```

Integration with Splunk

How to achieve basic integration between Diffusion and the Splunk™ analysis and monitoring application

About

Splunk is a third-party application from Splunk, Inc., which provides monitoring and analysis of other applications, primarily by parsing their logs and extracting information of interest. The information is displayed through a web interface, which allows the creation of dashboards and alerts on user-defined events. Splunk is available for all major operating systems.

The Diffusion log format is designed to be consistent and to allow for easy parsing by monitoring tools, not limited to Splunk.

Installation

Installation typically takes just a few minutes, see the appropriate section of the Splunk Installation Manual. For simplicity, we assume that Diffusion and Splunk are installed on the same machine.

Basic configuration

This is easier to do with existing log files to import, so configure Diffusion to write log files. To better demonstrate Splunk, set the server log file to TRACE logging in `etc/Logs.xml` and start Diffusion.

```
<!-- Example server log configuration -->
<log name="server">
 <log-directory>../logs</log-directory>
 <file-pattern>%s.log</file-pattern>
 <level>TRACE</level>
 <xml-format>false</xml-format>
 <file-limit>0</file-limit>
 <file-append>false</file-append>
 <file-count>1</file-count>
 <rotate-daily>false</rotate-daily>
</log>
```

On startup, access the Splunk web UI at http://localhost:8000. After logging in (and changing the default admin password), choose the **Add data** option.

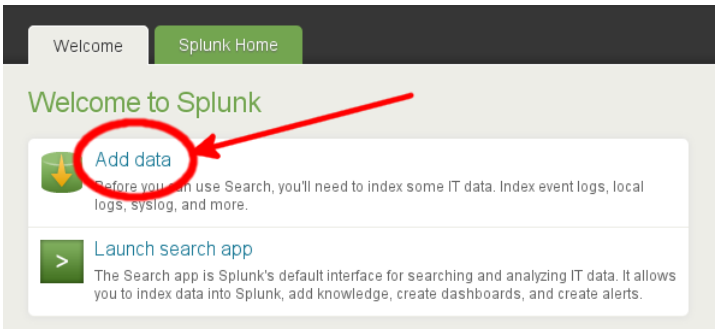
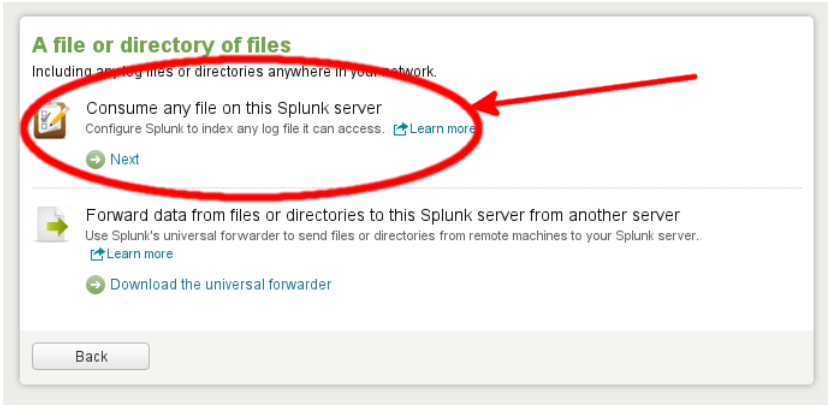


Figure 32: Welcome tab of the Splunk web UI

In the **Add Data to Splunk** screen that follows, choose the link **A file or directory of files** followed by **Consume any file on this Splunk server**.



Splunk might not be able to immediately identify the format of the log files; if this is the case, a dialog box similar to the following is presented. Select **csv** from the existing source types. Diffusion uses a pipe symbol rather than a comma as a separator but this is acceptable to the Splunk CSV parser.

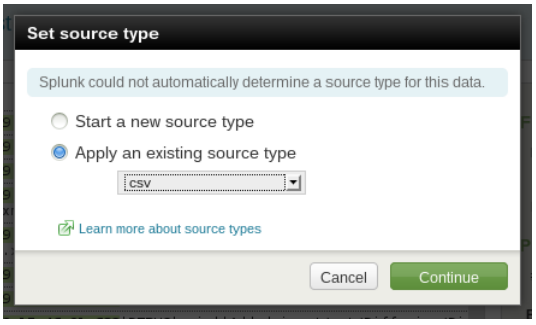


Figure 33: The Splunk Set source type dialog

The next dialog allows you to select the Diffusion `logs/Server.log` file under the **Preview data before indexing** option, which Splunk reads and parses. On the **Data Preview** screen, there are numbered log entries with the timestamp highlighted. This indicates that the log file has been correctly parsed. Accept this, and on the next screen, set the source to be continuously indexing the data. You can leave the parameters in **More settings** at their default values. Once this is done, you have given the new data source a name (for example, Diffusion Server Log) and finally accepted the settings, you can begin searching and generating reports based on the log contents.

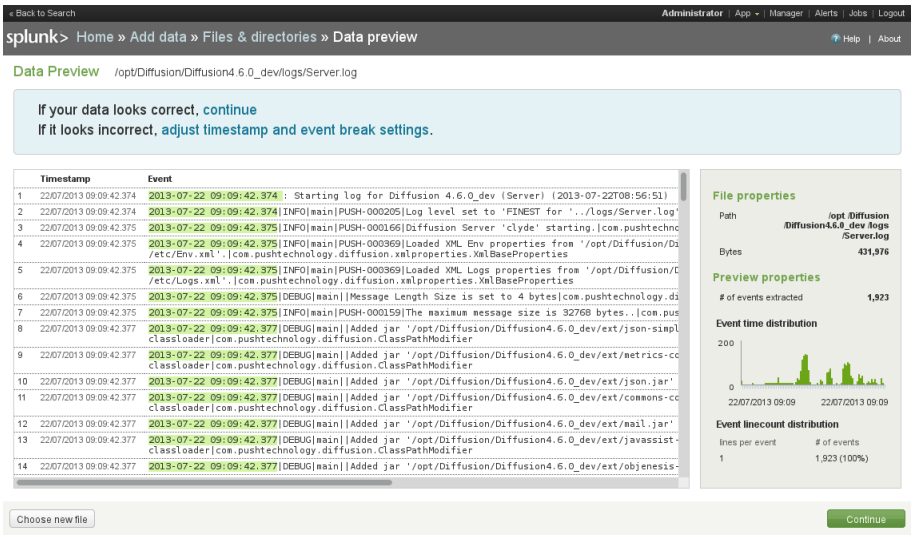


Figure 34: The Data Preview panel

Simple searches

Now we have a data source configured, we can start to execute basic searches.

On the Splunk launch page, select the **Search** option. On the **Search Summary** page that opens, select the Source relating to the file `logs/Server.log` previously imported. The page changes to include the source in the Search area. Additional search terms can be added to the end, for example, “Started Publisher”.

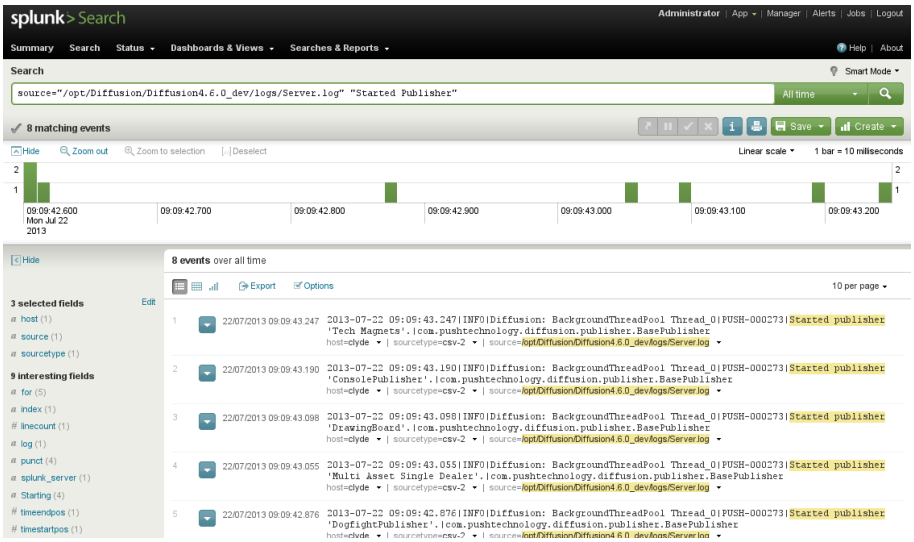


Figure 35: The Splunk search summary panel

Related concepts

JMX on page 511

You can use JMX to manage Diffusion. By default, the RMI registry port is 1099 and the JMX service port is 1100.

Related reference

Metrics on page 531

Diffusion metrics provide information about the server, client sessions, topics and log events. Diffusion can provide metrics in three main ways: via the web console, via JMX-compatible MBeans and via Prometheus.

Diffusion management console on page 538
A web console for managing the Diffusion server.

Logging on page 542
Diffusion uses the Simple Logging Facade for Java (SLF4J) API to log messages from the Diffusion server. SLF4J separates the logging of messages in the Diffusion server from the logging framework. This separation enables you to configure an independent back-end implementation to format and write out the log messages.

Related information

http://docs.splunk.com

Web servers

Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Related concepts

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Running the Diffusion server inside of a third-party web application server on page 621
Diffusion can run as a Java servlet inside any Java application server.

Hosting Diffusion web clients in a third-party web server on page 620
Host Diffusion web clients on a third-party web server to enable your customers to access them.

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Configuring Diffusion web server security on page 473
When configuring your Diffusion web server, consider the security of your solution.

Related reference

WebServer.xml on page 474
This file specifies the schema for the web server properties.

Diffusion web server

Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Any Diffusion connector can be configured to act as a web server and provide the following capabilities:

- Providing an endpoint for the HTTP-based transports used by Diffusion clients
- Hosting the Diffusion server landing page

- Hosting the Diffusion demos
- Hosting the Diffusion management console
- Optionally, hosting a static page you can use the check the status of the Diffusion server

For more information about configuring the Diffusion web server for these uses, see Configuring the Diffusion web server on page 472.

Related concepts

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Running the Diffusion server inside of a third-party web application server on page 621
Diffusion can run as a Java servlet inside any Java application server.

Hosting Diffusion web clients in a third-party web server on page 620
Host Diffusion web clients on a third-party web server to enable your customers to access them.

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Configuring Diffusion web server security on page 473
When configuring your Diffusion web server, consider the security of your solution.

Related reference

WebServer.xml on page 474
This file specifies the schema for the web server properties.

Server-side processing

A basic level of server-side processing can be utilized with any file hosted on the Diffusion web server that has a text mime type and JavaScript.

There are three server-side tags: Include, Publisher and Topic Data. These tags are stored in HTML comments so as to not interfere with normal HTML.

Include Tag

Include stubs load the file specified in the file attribute and are loaded as is into the parent HTML document. They do not necessarily have to be valid HTML. They can be positioned anywhere within the HTML file.

These includes are synonymous with `#Include` statements of ANSI C.

Below is an example of the syntax:

```
<!--@DiffusionTag type="Include" file="stub.html" -->
```

Include files can be nested so an include file can contain an include tag

Publisher tag

Publisher tags enable a publisher to interact with the web page during the serving process. Again these tags can appear anywhere within the HTML document. In the case below the publisher method `processHTMLTag` of the `Trade` publisher is called with the tag argument of `table` The publisher can return a String of HTML that is inserted into the document at the position of the tag and the tag is

removed. The `processHTMLTag` method is also called with the HTTP Request, although the request cannot be written to. Below is an example of the syntax

```
<!--@DiffusionTag type="publisher" publisher="Trade" tagid="table" --
>
```

TopicData

Topic data tags allow for `String`, `Int64` and `Double` items to be rendered in the HTML page. Again these tags can appear anywhere within the HTML document. The following example shows the syntax:

```
<!--@DiffusionTag type="TopicData" name="Assets/FX/EURUSD/O" -->
```

HTTP listener

Publishers can listen to all file HTTP requests by registering as a `HTTPRequestListener`. This exposes the interface

```
void handleHTTPRequest(HTTPVirtualHost virtualHost,HTTPRequest
 request)
```

This enables for more detailed statistics to be captured from the HTTP request

Hosting a status page on the Diffusion web server

You can host a simple status page on the Diffusion.

When setting up your Diffusion server to act as a web server for a status page, ensure that the web service uses a different connector to the Diffusion clients. This enables the web server to use a different thread pool and ensures that requests for status are not slowed by heavy client traffic.

You can use server-side tags to include topic data or call on publisher methods from within the status page. For more information, see Server-side processing on page 619.

Receiving no response from the status page might not indicate that the server hosting it is down. If you use a non-response from the status page as an indicator for failing over to another Diffusion server, ensure that you kill all processes belonging to the non-responsive Diffusion server before failing over.

Hosting Diffusion web clients in a third-party web server

Host Diffusion web clients on a third-party web server to enable your customers to access them.

If your Diffusion clients are web clients, they must be hosted on a web server to enable your customers to access them. We recommend that you use a third-party web server to host your clients instead of the built-in web server provided by Diffusion.

This approach requires additional configuration of your solution to account for cross-origin requests.

Cross-origin requests

Cross-origin requests occur when your web client requests resources (for example, data from the Diffusion server) that are hosted on a different domain, or in some cases a different port on the same domain, to your web client.

Some browsers do not support cross-origin resource sharing. For more information, see Cross-origin resource sharing limitations on page 40.

You can use one of the following approaches to enable cross-origin requests for your solution:

- Define a cross domain policy. For more information, see Cross domain policies on page 625.
- Use a load balancer to composite the URL spaces.

Related concepts

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Running the Diffusion server inside of a third-party web application server on page 621
Diffusion can run as a Java servlet inside any Java application server.

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Running the Diffusion server inside of a third-party web application server

Diffusion can run as a Java servlet inside any Java application server.

When running the Diffusion server inside a third-party web application server, the Diffusion server can have a different port number to clients that are hosted on the same server. This can cause cross-origin .

Some browsers do not support cross-origin resource sharing. For more information, see Cross-origin resource sharing limitations on page 40.

You can use one of the following approaches to enable cross-origin requests for your solution:

- Define a cross domain policy. For more information, see Cross domain policies on page 625.
- Use a load balancer to composite the URL spaces.

When using a third-party web server at least some of the functionality of the built-in Diffusion web server can be disabled. The file-service and http-service entries can be removed as Tomcat™ provides this functionality. The client-service is needed to support WebSocket and HTTP connection protocols. If these are not used, the client-service can be disabled as well.

Related concepts

Web servers on page 618
Diffusion incorporates its own basic web server for a limited set of uses. The Diffusion server also interacts with third-party web servers that host Diffusion web clients. The Diffusion server is also capable of being run as a Java servlet inside a web application server.

Diffusion web server on page 618
Diffusion incorporates its own web server. This web server is required to enable a number of Diffusion capabilities, but we recommend that you do not use it to host your production web applications.

Hosting Diffusion web clients in a third-party web server on page 620

Host Diffusion web clients on a third-party web server to enable your customers to access them.

Configuring the Diffusion web server on page 472
Use the `WebServer.xml` and `Aliases.xml` configuration files to configure the behavior of the Diffusion web server.

Web servers on page 130
Consider how to use web servers as part of your Diffusion solution.

Example: Deploying the Diffusion server within Tomcat

Run the Diffusion server inside Tomcat as a Java servlet.

About this task

The Tomcat servlet container and the Diffusion server run in the same Java process and can communicate directly through shared memory. Tomcat and the Diffusion server listen on different ports. Clients can connect directly to the Diffusion server without going through Tomcat.

Procedure

1. Configure an installation of the Diffusion server for how you want your Diffusion servlet to behave.

   Ensure, when editing the configuration files in the `etc` directory, that all paths are expressed as absolute paths.

   Ensure that a valid license file is present in the `etc` directory.

   Place any additional JARs that are required by your servlet in the `ext` directory of your Diffusion installation.
2. Use the `war.xml` Ant script in the `tools` directory of your Diffusion to package the Diffusion server as a WAR file.

   ```
   ant -f war.xml
   ```

   The script creates the `diffusion.war` file in the `build` directory of your Diffusion installation.

   The `diffusion.war` file includes the following files and directories:

   `META-INF/manifest.xml`
   : The manifest file for the WAR

   `WEB-INF/web.xml`
   : This file contains information about the servlet.

   `WEB-INF/classes`
   : This directory contains the configuration files for the Diffusion server. These files are copied from the `etc` directory of the Diffusion installation.

   `WEB-INF/lib/diffusion.jar`
   : The `diffusion.jar` file contains the Diffusion server

   `WEB-INF/lib`
   : This directory also contains JAR files copied from the `ext` directory of the Diffusion installation.

   `WEB-INF/lib/thirdparty`
   : This directory contains the third-party libraries that are required by the Diffusion server. These files are copied from the `lib/thirdparty` directory of the Diffusion installation.

   `lib/DIFFUSION`

This directory contains the browser API libraries. These files are copied from the `html/lib/DIFFUSION` directory of the Diffusion installation.

Additional files and directories

The WAR file contains additional files and directories that are not listed here.

The top level of the WAR file contains resources that can be served by Tomcat.

3. Verify the WAR file.
   a) Check that the WAR structure is the same as described in the previous step and that all necessary files have been copied into the WAR structure.
   b) Check that the `WEB-INF/web.xml` file contains the following information.

```
<servlet>
    <servlet-name>Diffusion</servlet-name>
    <display-name>Diffusion Servlet</display-name>
    <servlet-
class>com.pushtechnology.diffusion.servlet.DiffusionServlet</
servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

The WAR is now ready to be deployed inside a Java web application server. The rest of this task describes how to run the WAR inside of Tomcat, but you can use any Java web application server.

4. Define the Tomcat connectors for incoming connections in the `Server.xml` file.

   A connector defines the port, protocol, and various properties of how the connection is handled. The following is an example connector for handling HTTP 1.1 connections on port 8080:

```
<Connector port="8080"
        connectionTimeout="20000"
        URIEncoding="UTF-8"
        maxThreads="3"
        protocol="HTTP/1.1" />
```

   See the Tomcat documentation for more information.

5. When starting Tomcat, ensure that the following parameters are set:
   a) Set the `diffusion.home` parameter to the path to the Diffusion JAR file.

      `-Ddiffusion.home=`*`diffusion_installation`*`/lib`

      Tomcat must be aware of Diffusion.
   b) Set the `java.util.prefs.userRoot` parameter to a directory that Tomcat can write to.

      For example:

```
-Djava.util.prefs.userRoot=/var/lib/tomcat/diffusion/prefs/user
```

      Diffusion uses the `java.utils.prefs` mechanism to store preference information. If Tomcat, does not set this parameter, the Diffusion server logs warning messages.

What to do next

Accessing publishers from Tomcat

Diffusion started within Tomcat allows Tomcat to access the publishers. Tomcat can be used to serve JSP files providing dynamically generated content. These files can access publishers using the publishers class static methods.

```
<%@ page import="java.util.List,com.pushtechnology.api.publisher.*"
 %>
<html>
```

```
    <head>
        <title>Publisher Information</title>
    </head>
    <body>
        <table>
            <tr>
                <th>Publisher Name</th>
                <th>Topics</th>
            </tr>
            <% for (Publisher pub : Publishers.getPublishers()) { %>
            <tr>
                <td><%= pub.getPublisherName() %></td>
                <td><%= pub.getNumberOfTopics() %></td>
            </tr>
            <% } %>
        </table>
    </body>
</html>
```

The above is the content of a JSP file that return a list of the publisher Diffusion is running with the number of topics each publisher owns.

Other considerations when running the Diffusion server inside of a third-party web application server

Diffusion can run as a Java servlet inside any Java application server.

Apache Mod Proxy installation

Apache Mod Proxy can be used to forward HTTP requests from an Apache web server to Diffusion. It does not support persistent connections orWebSocket so the WebSocket connections do not work. Make sure that you include the following into the Apache configuration file (Virtual host setting).

```
ProxyPass /diffusion/ http://localhost:8080/diffusion/
```

For more information, see the Apache Mod Proxy documentation.

Apache AJP13 Installation

Apache AJP can be used to forward requests from an Apache web server to Tomcat. In the Apache virtual host configuration, mount the path

```
JkMount /diffusion/*dfnjetty
```

Workers definition file

```
worker.dfnjetty.port=8009
worker.dfnjetty.host=(host IP)
worker.dfnjetty.type=ajp13
worker.dfnjetty.lbfactor=1
worker.dfnjetty.cachesize=50
worker.dfnjetty.socket_keepalive=1

worker.list=dfnjetty
```

A connector that handles the AJP/1.3 protocol is needed running on port 8009 (because of the Workers file described above). See the Tomcat documentation for more information on this.

IIS Installation

Use an ISAPI_Rewrite tool. For example, http://www.helicontech.com/isapi_rewrite

The rewrite rule is as follows:

```
RewriteEngine on RewriteRule ^diffusion/ http://localhost:8080/
diffusion/ [p]
```

Diffusion home directory

The servlet container must be aware of Diffusion. Add the path to the directory that contains the Diffusion JAR file to the Java VM arguments that you use to start the servlet container.

`-Ddiffusion.home=`*`diffusion_installation`*`/lib`

Cross domain policies

Cross domain policies grant permission to communicate with servers other than the one the client is hosted on.

Cross-domain XML file

The cross-domain policy is defined in an XML file

A cross-domain policy file is an XML document that grants a web client permission to handle data across multiple domains. When a client hosts content from a particular source domain and that content makes requests directed towards a domain other than its own, the remote domain must host a cross-domain policy file that grants access to the source domain, allowing the client to continue with the transaction. Policy files grant read access to data, permit a client to include custom headers in cross-domain requests, and are also used with sockets to grant permissions for socket-based connections.

For example, say that the Diffusion client is loaded from static.example.com and the connection URL to the Diffusion client is `http://streaming.example.com`, a `crossdomain.xml` file must be loaded from `static.example.com`

A `crossdomain.xml` is required if one of the following is true:

- You are using Diffusion as a streaming data server and a separate web server which are on different domains
- The Diffusion connection type is HTTP or HTTPS
- You are not using a load balancer to HTTP rewrite Diffusion traffic

> **Note:** You cannot use the iframe streaming transport with cross-domain requests. This is not supported by Diffusion.

Load balancers

Load balancers provide many capabilities that are key to creating a seamless Diffusion solution. We recommend that you use a load balancer with Diffusion.

In addition to balancing client connections across multiple Diffusion servers, you can use load balancers to composite URL spaces or do SSL offloading.

Connections between Diffusion clients and Diffusion servers have specific requirements. If your load balancer handles Diffusion connections incorrectly, this can cause problems for your solution.

Ensure that you fully review this section of the user guide when using load balancers with Diffusion.

Routing strategies at your load balancer

Your load balancer can present a number of different strategies for choosing which Diffusion server a new client connection is routed to. After a client connection has been routed to a Diffusion server, ensure that the client is always routed to the Diffusion server that its session exists on.

The routing strategies that are available to you depend on the load balancer that you choose to use. The following table lists some examples of routing strategies:

Table 50: Examples of routing strategies

Name	Description
Round-robin	Each available Diffusion server is chosen in turn, with none favored.
Fewest clients	The Diffusion server with the fewest number of client connections in progress is chosen.
Least loaded	The Diffusion server with the lowest CPU load is chosen.

Routing connection that use HTTP protocols

To route HTTP traffic, the load balancer must be able to inspect the HTTP headers and extract session information.

Diffusion sets an HTTP cookie named `session` with a connection-specific ID specifically for this purpose. Your load balancer can use this cookie to maintain a table of client to server mappings.

The session cookie is flagged with `HttpOnly`, which prevents scripts accessing the cookie. The session cookie is not flagged with `Secure`, which prevents the cookie from being sent over non-secure connections.

Sample HTTP conversation, cookie highlighted:

```
POST /diffusion/ HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 0
tt: 90
Origin: http://localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
 (KHTML, like Gecko) Ubuntu Chromium/48.0.2564.82 Chrome/48.0.2564.82
 Safari/537.36
m: 0
ty: B
v: 4
Content-Type: text/plain;charset=UTF-8
Accept: */*
Referer: http://localhost:8080/tools/DhtmlClient.html
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6

HTTP/1.1 200 OK
Set-Cookie: session=c04815df73a1646d-0000000000000000; HttpOnly
Access-Control-Allow-Origin:http://localhost:8080
Cache-Control:no-store, no-cache
Content-Type:text/plain; charset=UTF-8
Content-Length:41

4.100.4.c04815df73a1646d-0000000000000000
```

```
POST /diffusion/ HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 0
Origin: http://localhost:8080
c: c04815df73a1646d-0000000000000000
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
 (KHTML, like Gecko) Ubuntu Chromium/48.0.2564.82 Chrome/48.0.2564.82
 Safari/537.36
m: 1
Content-Type: text/plain;charset=UTF-8
Accept: */*
Referer: http://localhost:8080/tools/DhtmlClient.html
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6
Cookie: session=c04815df73a1646d-0000000000000000
```

Diffusion uses a session ID for the cookie. This enables the load balancer to maintain a map of each session ID to its target Diffusion server.

Instead, you can disable the Diffusion cookie and configure the load balancer to set a cookie that identifies the target server instead of the session. While the overhead of transmitting a cookie is still present between client and load balancer, the identifier can be smaller because there are a smaller number of servers than client sessions. Load balancers that use the cookie to identify the Diffusion server can send less data down the wire and consume fewer resources.

If you want to configure your load balancer to inject its own cookie, you can disable this Diffusion cookie. To disable the Diffusion cookie, set the `<disable-cookie>` element in the `WebServer.xml` configuration file of your Diffusion to `true`.

Routing connections that use streaming protocols

Streaming protocols that open a single socket and remain connected until they are no longer required appear immune to requiring any special routing considerations. However, in the event that connection keep-alive is enabled to handle reconnections in case of temporary connection loss, it is important that the reconnection attempt is routed to the original server.

Without the ability to parse headers (and indeed, the absence of a session ID at all), the most common method for routing a streaming protocol such as WebSocket is to create a client/server mapping based on the IP addresses of the endpoints. This technique is generally referred to as Sticky-IP, and has the advantage of also working with HTTP transports, if required.

For F5®'s Sticky IP, ensure that the Source Address Translation option is set to Auto Map.

Figure 36: Sticky-IP in F5 BIG-IP

The drawback of this approach is that multiple users masquerading behind a proxy or access point can have the same IP address, and all requests from clients with that IP address are routed to the same Diffusion instance. Load balancing still occurs, but some hosts might be unfairly loaded.

Monitoring available Diffusion servers from your load balancer

To route your client connections most effectively, your load balancer must know which Diffusion servers are available to accept connections.

There are a number of ways to determine the availability of a Diffusion server:

- Implement a custom monitor using a scripting language that is supported by your load balancer.

  For example, create a custom Diffusion client that connects and subscribes to a status topic.

  This is the most effective way of determine availability as can check the connector used by your client applications.
- Use an HTTP probe against the built-in web server.

  This has the advantage of being simple; most system administrators are familiar with HTTP requests. In the simplest case, a GET request can be made against the root context of the web server, for example:

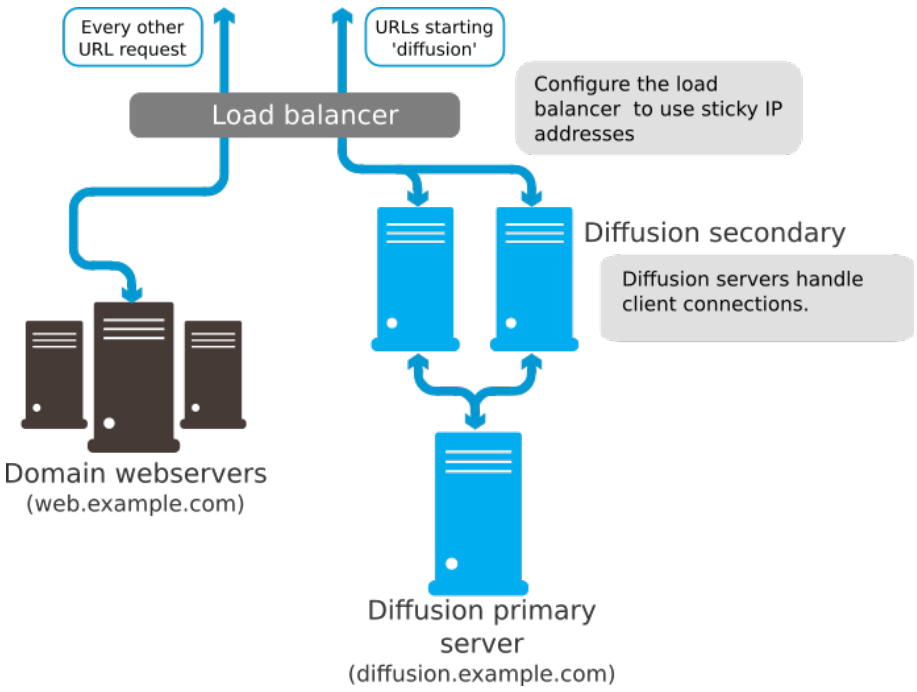
  ```
  GET / HTTP/1.0\r\n
  ```

  However, this only tests the availability of the Diffusion server as a whole, and not the applications within it.

Compositing URL spaces using your load balancer

If your Diffusion servers are located at a different URL to the Diffusion browser clients hosted by your web servers, you can use a load balancer to composite the URL spaces.

Security features in some browsers prevent web-based Diffusion clients from making requests to your Diffusion server if your Diffusion server is in a different URL space to the web server you use to host your client.



- Your web content is hosted on `web.example.com` and your Diffusion servers are hosted on `diffusion.example.com`.

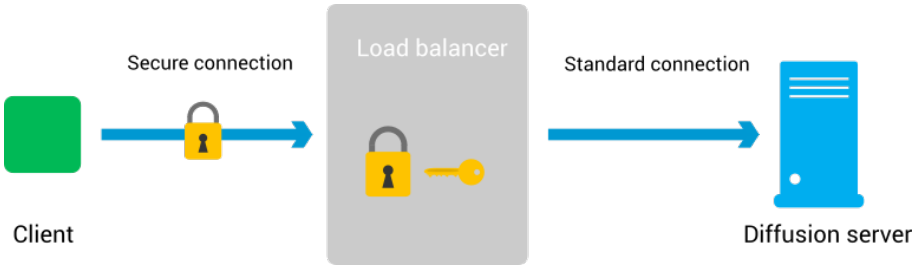
- The load balancer composites the URL space so that requests to Diffusion at `web.example.com` are routed to Diffusion servers hosted on `diffusion.example.com`
- To the client, both the web and the Diffusion content appear to be hosted on `web.example.com`. This avoids any cross-domain security issues.

If you choose not to use your load balancer to composite the URL spaces, you can set up cross-domain policy files that allow requests to the different URL spaces.

Secure Sockets Layer (SSL) offloading at your load balancer

Diffusion clients can connect to your solution using TLS or SSL. The TLS/SSL can terminate at your load balancer or at your Diffusion server.

> **Note:** If you have sensitive client data that you must secure, use a secure connection all the way from the client to the Diffusion server. Either do not perform SSL offloading at the load balancer or re-encrypt the connection between load balancer and the Diffusion server.



SSL offloading is when you terminate the TLS at the load balancer. The processing burden of encrypting and/or decrypting a Diffusion client connection made over SSL can then be is offloaded to a component that can perform SSL termination more efficiently.

After the SSL connection has been decrypted, the client connection can travel between the load balancer and the Diffusion server using an unsecured transport. Doing this reduces CPU cost on your Diffusion servers.

Using load balancers for resilience

An important part of creating your Diffusion solution is ensuring that it is resilient if one of its components fails.

Load balancer redundancy

If you only have one load balancer in your Diffusion solution, this load balancer can become a single point of failure. For a more resilient solution, have more than one load balancer.

You can have a cluster of active load balancers, each with a different IP address, behind a DNS that uses a round-robin strategy to direct client connections to a hostname to the IP of each load balancer in turn. With a round-robin routing strategy at your DNS, there can be a lag between a load balancer becoming unavailable and this being detected.

Alternatively, you can configure your network so that all of your load balancers are available on the same IP address.

If you use multiple load balancers, ensure that all load balancers have access to any client/server mapping information.

Refer to the documentation for your load balancer for further information about load balancer redundancy.

Using load balancers with Diffusion replication

Diffusion replication is designed to be used with Diffusion servers that are load balanced. If a Diffusion server in your solution becomes unavailable, your load balancer must re-route any of that server's client connections to other Diffusion servers in your solution.

For more information, see Server clusters for high availability on page 105.

Common issues when using a load balancer

There are some configuration options on your load balancer that can cause problems or inefficient behavior in your Diffusion solution.

Load balancer closing silent connections

Many load balancers have a default configuration that closes TCP connections after a few seconds of the connection being silent. This is appropriate for a load balancer handling connections to a web server. However, Diffusion traffic is different and there can be long intervals of silence on the connection.

Do not configure load balancers or firewalls to close TCP connections that are not transmitting data. Diffusion maintains a connection for every live session so that data can be pushed.

If a network device terminates a TCP connection autonomously, the Diffusion server might interpret this as a close initiated by the client and close the session. If this happens, any reconnection attempts made by the client fail.

Connection pooling

Many load balancers include a connection pooling feature where connections between the load balancer and the Diffusion server are kept alive and reused by other clients. In fact, multiple clients can be multiplexed through a single server-side connection.

In Diffusion, a client is associated with a single TCP/HTTP connection for the lifetime of that connection. If a Diffusion server closes a client, the connection is also closed. Diffusion makes no distinction between a single client connection and a multiplexed connection, so when a client sharing a multiplexed connection closes, the connection between the load balancer and Diffusion is closed, and subsequently all of the client-side connections multiplexed through that server-side connection are closed.

For this reason, it is required that load balancers are not configured to pool connections when working with Diffusion.

Reuse TCP connection

If your load balancer is configured to create a new TCP connection between the load balancer and the Diffusion server for each request from a specific client, this can be expensive. Creating a new TCP connection per request, increases the time each request takes to be processed and increases the amount of traffic between the load balancer and the Diffusion server.

To avoid this, ensure that your load balancer is configured to reuse a TCP connection for requests from the same client.

Sticky-IP

We recommend that you use the sticky-by-IP routing strategy when your clients connect using streaming protocols. This ensures that client connections are always routed to the Diffusion server where their sessions are located.

However, the drawback of this approach is that multiple users masquerading behind a proxy or access point can have the same IP address, and all requests from clients with that IP address are routed to the same Diffusion server. Load balancing still occurs, but some hosts might be unfairly loaded.

TCP retransmission timeout

If you use Diffusion failover, the TCP retransmission timeout on your load balancer's host server can cause long waits for clients whose connections failover from one Diffusion server to another. When a Diffusion server becomes unavailable, the load balancer can hold open existing client connections to this server. These connections can continue to receive and buffer data from the client for the duration of the timeout, before being closed. This data is discarded when the connection closes.

You can avoid this problem by changing the TCP retransmission timeout of the host server of your load balancer or by configuring the load balancer to shutdown connections to Diffusion servers it knows are unhealthy.

MQTT support

Diffusion supports MQTT clients connecting to the Diffusion server to send and receive topic data, with no extra code required on the client.

MQTT

Diffusion supports the MQTT 5.0 protocol. MQTT is a popular messaging standard for communicating with Internet of Things (IoT) devices.

MQTT clients can publish and subscribe to Diffusion topics.

MQTT is widely supported and works on low-powered devices. It is useful for communicating with devices that are unable to run a Diffusion client SDK, or in applications where you are unable to install code on remote devices.

Using Diffusion instead of a standard MQTT broker enables you to take advantage of advanced Diffusion features such as:

- Server-side data processing with topic views
- Monitoring and management using the Diffusion management console, JMX and/or Prometheus
- High-availability and scaling
- Diffusion's fine-grained security model
- Automatic topic removal and missing topic notifications
- Control of MQTT sessions from clients developed with the Diffusion SDKs

Using MQTT

For full details of the capabilities and limitations of MQTT support, see Using MQTT on page 632.

Configuring MQTT

MQTT support is enabled by default. For details of how to enable or disable MQTT, see Configuring MQTT on page 634.

Using MQTT

MQTT 5.0 clients can publish and subscribe to Diffusion topics and support is enabled by default.

Connecting via MQTT

By default, you can connect to a Diffusion server from any MQTT 5.0 client, as if you were connecting to a standard MQTT broker.

For a TCP connection, you will need to provide:

- Port: `8080` by default, or as configured in Connectors.xml
- Username and password: use a Diffusion principal and password (or configure Diffusion to allow anonymous connections on the topic path you require)

For a WebSocket connection, you will also need to provide:

- URL path: `diffusion`

Connect your MQTT client to the host name provided in your Cloud service dashboard (the same one that you would use to connect using an SDK client).

You will need to provide:

- Port: `443`
- Username and password: use a Diffusion principal and password (or configure Diffusion to allow anonymous connections on the topic path you require)
- URL path: `diffusion`

MQTT protocol support and limitations

Diffusion currently supports only MQTT 5.0. Earlier versions of the protocol (such as MQTT 3.1.1) are not supported.

MQTT clients can connect by either TCP or WebSocket.

Secure connections are supported. TLS support is the same as for SDK client connections, including TLS 1.3. See SSL and TLS support on page 38 for details.

MQTT wildcard filters are supported.

All topics are ordered topics.

Quality of service (QoS) support:

- Subscribing to topics with an MQTT client: QoS 0.
- Publishing from an MQTT client to Diffusion: QoS 0 or 1.

We recommend you use QoS 1 when publishing from MQTT. This enables Diffusion to notify the MQTT client if an update fails, and is also used for flow control.

Diffusion feature support

MQTT clients can publish and subscribe to Diffusion topics.

Diffusion security authorization rules are applied to publication and subscription via MQTT.

MQTT sessions will trigger automatic topic removal and missing topic notifications like other sessions.

Diffusion clients written with the SDK can control MQTT sessions.

MQTT clients cannot use:

- request-response messaging

- topic control, session control and server administration features
- recordV2 topics
- advanced time series topic features (subscribing to a time series and appending an event are supported)
- delta streaming
- topic compression
- reliable reconnection

These features are only supported by clients developed with a Diffusion SDK.

Session properties

MQTT sessions can be distinguished by their session properties:

- `$ClientType`: `MQTT`
- `$Transport`: `TCP` or `WebSocket`
- `$MQTTClientId` (new session property): the client identifier provided by the client; if this is empty, the `$SessionId` is used as a client identifier

Translating between MQTT and Diffusion

MQTT 5.0 clients can both publish and subscribe to Diffusion topics.

Paths containing the MQTT wildcard filter characters (# and +) or starting with $ are not valid MQTT topic names. MQTT clients cannot subscribe to Diffusion topics with such paths.

Table 51: Subscribing (updates from Diffusion to MQTT client)

Diffusion topic type	MQTT payload format	PUBLISH content type	Conversion	If conversion fails
binary	unspecified	not set	Publish value.	-
string	utf8	not set	Convert payload to UTF-8. Replace malformed-input and unmappable character sequences with ?.	-
double	utf8	application/json	Convert to decimal representation in ASCII, as per java.lang.Double.toString(double).	Log warning and send as unspecified payload.
int64	utf8	application/json	Convert to decimal representation in ASCII, as per java.lang.Long.toString(long).	Log warning and send as unspecified payload.
json	utf8	application/json	Convert CBOR to JSON string.	Log warning and send as unspecified payload.
recordv2	-	-	None - subscriptions will not resolve.	-
time series	unspecified or UTF-8	depends on event type	Send event value according to the rules above. MQTT subscribers are not sent event metadata (author, timestamp).	depends on conversion

Table 52: Publishing (updates from MQTT client to Diffusion)

Diffusion topic type	Conversion	If conversion fails
topic does not exist	Create topic (see below)	-
binary	Update topic with payload.	-
string	Convert payload to UTF-8, then CBOR. Replace malformed-input and unmappable character sequences with ?.	-
double	Interpret as decimal representation in ASCII and convert using java.lang.Double.valueOf(String), then CBOR.	Log warning and fail with 0x83 (implementation specific error).
int64	Interpret as decimal representation in ASCII and convert using java.lang.Long.valueOf(String), then CBOR.	Log warning and fail with 0x83 (implementation specific error).
json	Interpret as JSON string, then CBOR.	Log warning and fail with 0x83 (implementation specific error).
recordv2	Log warning and fail with 0x83 (implementation specific error).	-
time series	Convert payload according to the rules above. Append to the time series using the current principal and timestamp. MQTT publishers cannot provide event metadata nor edit events.	depends on conversion

Creating a Diffusion topic from MQTT

If an MQTT client publishes an update to a topic path which has no existing Diffusion topic, the topic is automatically created.

The Diffusion topic type created is determined as follows:

- if the MQTT PUBLISH packet contains a content type property of `application/json`, a JSON topic is created (this fails if the payload can't be parsed as JSON).
- otherwise, if the `payload format` property is `UTF-8`, a string topic is created
- otherwise, a binary topic is created

> **Note:** MQTT supports zero-length (empty) levels of a topic path but these cannot be published to Diffusion. For example, `a/b//c` (with an empty level between `b` and `c`) is a valid MQTT topic, but trying to use it to publish to Diffusion will fail.

Configuring MQTT

Use the `Connectors.xml` configuration file to enable or disable MQTT protocol support.

Edit the `protocols` element of `etc/Connectors.xml` to enable or disable MQTT support. The WebSocket and TCP transports can be enabled or disabled independently.

See Connectors.xml for more details.

Note that in a fresh Diffusion installation, MQTT is enabled by default on port 8080.

Kafka adapter

The Kafka adapter for Diffusion enables sharing messages between Diffusion and Apache Kafka clusters.

Kafka adapter

The Kafka adapter comprises the following files all located in the `adapters/kafka` directory of your Diffusion installation:

`kafka-adapter-6.7.7.jar`

This JAR file contains the Java application that links the Diffusion server and a Kafka cluster.

`application.conf`

This file is used to configure the Kafka adapter.

`configSchema.json`

This JSON schema file can be used to validate the `application.conf` configuration file.

`kafka_adapter.sh` and `kafka_adapter.bat`

These executable files can be used to start the Kafka adapter on UNIX, Linux, or Windows systems.

`log4j2.xml`

The Kafka adapter requires Java. The version requirements are the same as for the Diffusion server.

Using the Kafka adapter

To use the Kafka adapter, first configure it by editing the `application.conf` to define the adapter behavior. For more information, see Configuring the Kafka adapter on page 635.

To run the adapter, use the .sh or .bat file. See Running the Kafka adapter on page 638.

Monitoring the Kafka adapter

The adapter uses Prometheus to expose its metrics. This can be accessed using the `http://localhost:8000/metrics` endpoint. The default '8000' port can be changed in the configuration file.

The adapter can also be visualised and monitored in the Diffusion management console, using the `Network` tab.

Configuring the Kafka adapter

Use the `application.conf` configuration file to configure how the adapter shares data between Diffusion and Kafka.

Edit the application.conf JSON file to configure the adapter's behavior.

The "diffusion" section is used to configure the connection to Diffusion.

```
"diffusion": {
      // URL of Diffusion server.
      "url": "ws://localhost:8080",
```

```
      // Principal to be used to connect to Diffusion server.
      "principal": "admin",
      // Password to be used to connect to Diffusion server.
      "password": "password",
      // Map of user defined properties to pass to create session.
      // "properties": {"key":"value"},
      // Optional. Interval between Diffusion server reconnection
 attempts in milliseconds. Defaults to 1000.
      "reconnectIntervalMs": 1000,
      // Optional. Maximum message size to be used when creating
 Diffusion session. Defaults to Integer.MAX_VALUE.
      // "maximumMessageSize": 50000,
      // Optional. Maximum queue size to be used when creating
 Diffusion session.  Defaults to 10000.
      // "maximumQueueSize": 20000
      // Optional. Output buffer size (in bytes) to be used when
 creating Diffusion session. Defaults to 512KiB.
      // "outputBufferSize": 200000
      // Optional. Input buffer size (in bytes) to be used when
 creating Diffusion session. Defaults to 512KiB.
      // "inputBufferSize": 200000
      }
```

The "kafkaClusters" section defines which Kafka clusters to connect to.

```
  "kafkaClusters": [
 {
   // Unique identifier to identify this Kafka cluster.
   "clusterId": "localCluster",
   // List of host and port pairs that are addresses of the Kafka
 brokers in the Kafka cluster.
   "bootstrapServers" : ["localhost:9092", "localhost:9093"],
   // Optional. If used, SSL connection will be used to connect to
 Kafka brokers.
   // "sslDetails": {
       // Kafka client's JKS truststore's file location.
       // "trustStoreLocation": "/etc/kafka.client.truststore.jks",
       // Kafka client's JKS truststore's password.
       // "trustStorePassword": "clientpass"
   // },
   // Optional. If used, SASL authentication is enabled. Contains
 configuration details for SASL authentication.
   // "saslDetails": {
       // Allowed values are PLAIN, GSSAPI.
       // "saslMechanism": "GSSAPI",
       // Mandatory, if saslMechanism is GSSAPI.
       // "kerberosServiceName": "kafkaAdapter"
       // Mandatory, if saslMechanism is PLAIN.
       // "userName": "admin",
       // Mandatory, if saslMechanism is PLAIN.
       // "password": "admin-secret"
   // }
 }
],
```

The "publisher" section defines how to consume Kafka topics and publish to Diffusion.

```
  "publisher": {
"subscriptions": [
  {
    // Kafka cluster id defined in kafka.clusterId field.
    "kafkaClusterId": "localCluster",
```

```
    // Optional. Map of additional Kafka consumer configuration's
 keys and value pairs.
    "kafkaConsumerConfigurations": {"auto.offset.reset":
 "latest","metadata.max.age.ms": 1000},
    // A unique string that identifies the consumer group for Kafka
 consumers created for this cluster.
    "consumerGroupId": "diffusionKafkaAdapter",
    // Optional. Number of Kafka consumers to be created per key-
value type pair for this cluster. Defaults to 1.
    "consumerCount": 3,
    // Optional. Corresponds to "max.poll.interval.ms" configuration
 field of Kafka consumer. Defaults to 300 seconds.
    "consumerPollTimeoutMs": 1000,
    // Optional. List of Kafka topic's regular expression details to
 subscribe to in this cluster.
    "regexSubscriptions": [
      {
        // Kafka topic regular expression to subscribe to.
        "name": "fx.*",
        // Data type of 'key' for this regular expression
 subscription.
        // Allowed values are: BYTEARRAY, BYTEBUFFER, BYTES, DOUBLE,
 FLOAT, INTEGER, LONG, SHORT, STRING, UUID, JSON.
        "keyType": "STRING",
        // Data type of 'value' for this regular expression
 subscription.
        // Allowed values are: BYTEARRAY, BYTEBUFFER, BYTES, DOUBLE,
 FLOAT, INTEGER, LONG, SHORT, STRING, UUID, JSON.
        "valueType": "JSON",
        // Optional. Flag to specify whether to map to timeseries
 topic. Defaults to false.
        // "mapToTimeSeriesTopic": true,
        // Optional. Prefix to be used when creating Diffusion topic.
 Defaults to "".
        "diffusionTopicPrefix": "kafka/",
        // Optional. Map of string key-value pairs of Diffusion topic
 properties which will be used during topic creation.
        "diffusionTopicProperties": {"DONT_RETAIN_VALUE":"true",
"CONFLATION":"off"}
      }
    ]
    //Optional. List of Kafka topic details to subscribe to in this
 cluster.
    "topicSubscriptions": [
      {
        // Kafka topic name to subscribe to.
        "name": "sampleTopic",
        // Data type of 'key' for this topic subscription.
        // Allowed values are: BYTEARRAY, BYTEBUFFER, BYTES, DOUBLE,
 FLOAT, INTEGER, LONG, SHORT, STRING, UUID, JSON.
        "keyType": "STRING",
        // Data type of 'key' for this topic subscription.
        // Allowed Values are: BYTEARRAY, BYTEBUFFER, BYTES, DOUBLE,
 FLOAT, INTEGER, LONG, SHORT, STRING, UUID, JSON.
        "valueType": "STRING",
        // Optional. Flag to specify whether to map to timeseries
 topic. Defaults to false.
        // "mapToTimeSeriesTopic": true,
        // Optional. Prefix to be used when creating Diffusion topic.
 Defaults to "".
        "diffusionTopicPrefix": "kafka/",
        // Optional. Map of string key-value pairs of Diffusion topic
 properties which will be used during topic creation.
```

```
        // "diffusionTopicProperties": {"DONT_RETAIN_VALUE":"true",
 "CONFLATION":"off"}
      }
    ]
  }
],
 // Optional. Prefix to be used when creating Diffusion topics. This
 prefix will be applied for all Diffusion topics created. Defaults to
 "".
 // "commonDiffusionTopicPrefix": "kafka/",
 // Optional. Map of string key-value pairs of Diffusion topic
 properties which will be used during topic creation.
 // These properties will be applied for all Diffusion topics
 created.
 // "commonDiffusionTopicProperties": {"DONT_RETAIN_VALUE":"true",
 "CONFLATION":"off"}
},
```

The "subscriber" section defines how to subscribe to Diffusion topics and send updates to Kafka.

```
  "subscriber": {
"subscriptions": [
  {
    "kafkaClusterId": "localCluster",
    "kafkaProducerConfigurations": {"client.id":
 "diffusionProducer"},
    "topicDetails": [
      {
        // Diffusion topic path or path selector to subscribe to.
        "diffusionTopic": "?kafka/.*//",
        // Data type of value to be received for this Diffusion topic
 path.
        // Allowed values are: JSON, BINARY, STRING, INT64, DOUBLE.
        "diffusionTopicType": "JSON",
        // Optional. If this field is set, this topic is used to
 send all Diffusion updates from the matching selector to the single
 specified Kafka topic.
        // "mapToSingleKafkaTopic": "godTopic",
        // Optional. If this field is set, Kafka topics are created
 based on the source Diffusion topic path with the specified prefix.
        "kafkaTopicPrefix": "diffusion.",
        // Optional. Kafka key value to be used to send Kafka update.
 Defaults to "1".
        "kafkaKey": "1"
      }
    ]
  }
]
},
```

You can delete or comment out the "publisher" or "subscriber" section if you only need one-way functionality. You can leave both enabled if you want the adapter to work both ways at once.

Running the Kafka adapter

The Kafka adapter is not enabled by default.

The Kafka adapter is a Java application. The adapter has the same version requirements as the Diffusion server.

To run the Kafka adapter, complete the following steps:

1. Use the `application.conf` configuration file to define the Kafka adapter behavior.

   For more information, see Configuring the Kafka Adapter.
2. Run the `kafka_adapter.sh` or `kafka_adapter.bat` script to start the Kafka adapter:

```
./kafka_adapter.sh
```

To set up a secure connection with Diffusion:

1. Import the Certificate Authority certificate which is used to sign the server's certificate to the local truststore: `keytool -keystore local.client.truststore.jks -alias <alias> -import -file <ca-cert file name> -storepass <password> -keypass <password> -noprompt`
2. Add the following system properties as parameters to the `java` command in your .sh or .bat start script: `-Djavax.net.ssl.trustStore=./local.client.trustStore.jks -Djavax.net.ssl.trustStorePassword=<password>`
3. Update the URL of the Diffusion server in `application.conf` to use "wss://"

JMS adapter

The JMS adapter for Diffusion, enables Diffusion clients to transparently send data to and receive data from destinations (topics and queues) on a JMS server.

JMS adapter

The JMS adapter comprises the following files all located in the `adapters/jms` directory of your Diffusion installation:

`jmsadapter-6.7.7.jar`
: This JAR file contains the Diffusion Java application that links the Diffusion server and a JMS server.

`configuration.json`
: This file is used to configure the JMS adapter.

`configSchema.json`
: This JSON schema file can be used to validate `configuration.json`.

`README.txt`
: This file contains details on how to run the adapter.

`jms_adapter.sh` and `jms_adapter.bat`
: These executable files can be used to start the JMS adapter when running it as a standalone client on UNIX, Linux, or Windows systems.

The JMS adapter can be run as a client on any system that has a Java 8 JRE installed on it - Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later) is recommended.

Using the JMS adapter

To use the JMS adapter, first configure it by editing `configuration.json` to define the adapter behavior. For more information, see Configuring the JMS adapter on page 640.

To run the adapter, use the .sh or .bat file. For more information, see Running the JMS adapter on page 640.

Configuring the JMS adapter

Use the `configuration.json` configuration file to configure the JMS adapter to send and receive messages with destinations on a JMS server.

Edit the configuration.json JSON file to configure the adapter's behavior.

Refer to the `configSchema.json` for details of the configuration options.

Related concepts

JMS on page 133
Consider whether to incorporate JMS providers into your solution.

Transforming JMS messages into Diffusion messages or updates
Publishing using the JMS adapter

Running the JMS adapter

The JMS adapter is not enabled by default.

The JMS adapter is a Java application. The adapter requires Java 8 (8u131-b11 GA or later) or Java 11 (11.0.3 GA or later).

To run the JMS adapter, complete the following steps:

1. Copy the client JAR file for the third-party JMS provider into the `adapters/jms` directory within your Diffusion installation directory.
2. Copy any required SLF4J binding JARs (for example `slf4j-simple-1.7.30.jar`) if not included in the JMS provider's client JAR. Some SLF4J binding JARs are also available under `lib/thirdparty/` in your installation directory. Add any necessary logging configuration file.
3. Use the `configuration.json` configuration file to define the JMS adapter behaviour. For more information, see Configuring the JMS adapter on page 640.
4. Edit the `jms_adapter.sh` or `jms_adapter.bat` file to include the path to the JMS provider client JAR and SLF4J JARs on the classpath.
5. Run the `jms_adapter.sh` or `jms_adapter.bat` script with the relative path to `configuration.json` as an argument: `jms_adapter.sh configuration.json`

CDC database adapter

The CDC adapter for Diffusion enables sharing data between Diffusion and a compatible database.

CDC adapter

The CDC (Change Data Capture) adapter can be used to connect a Diffusion server (or server cluster) to a database server. It uses Debezium (1.6.1.Final), under the hood to connect with databases.

The adapter can fetch the latest data, with its schema, from multiple databases and publish to JSON Diffusion topics. Any data change events captured by the Debezium connector are processed and published to Diffusion topics according to the configuration.

Since the adapter uses Debezium as a solution to capture the data change events, all configuration options supported by Debezium are allowed. Please refer to the Debezium documentation at https://debezium.io/documentation/ to understand the supported options.

The CDC adapter comprises the following files all located in the `adapters/cdc` directory of your Diffusion installation:

`cdc-adapter-6.7.7.jar`

This JAR file contains the Java application that links the Diffusion server and a database.

`configuration.json`

This file is used to configure the CDC adapter.

`configSchema.json`

This JSON schema file can be used to validate the `configuration.json` configuration file.

`log4j2.xml`

Log file.

The CDC adapter requires Java. The version requirements are the same as for the Diffusion server.

The CDC adapter is tested with MySQL and PostgreSQL. Connection to other databases may work but is not supported.

Services

A service represents a specific function that this adapter supports. This can be of a specific type which is identified by a unique name, specific to adapters.

Each service has a type, name, description and configuration. The overall configuration of the adapter is defined by each service configuration. The services can be added/updated/removed via the Diffusion console, after the adapter is started.

If the adapter is started with a complete configuration file, these configurations will be used to create services, during adapter startup, which will be visible in the console.

The CDC adapter supports two types of service which are required to set up required interactions between databases and Diffusion.

Database Connector Service

The Database Connector Service is used to define Debezium connector configuration for a specific database, in the adapter. Once this service is defined, it can be referred to in the configuration of the Diffusion publisher service. This means that any common Debezium connector configuration can be defined as a Database connector service, and reused in multiple Diffusion publisher services.

Adding this service will not start any process, but will update the overall configuration of the adapter to contain the Debezium connector configuration detail.

This service does not support any operations.

Diffusion Publisher Service

A Diffusion publisher service is used to start the Debezium connector engine, and begin streaming changed data values from the database to Diffusion topics. There can be multiple instances of this service added to the adapter, to consume data from tables in the database, using different configuration.

Using the CDC adapter

Before you run the adapter, you can configure it by editing the `configuration.json` to define the adapter behavior. You can also pass a bootstrap configuration as a system property, or you can combine system properties and the configuration file. For more information, see Configuring the CDC adapter on page 642.

Configuration can also be updated during runtime via the Diffusion console.

To run the adapter, run the .jar file. See Running the CDC adapter on page 643.

Monitoring the CDC adapter

The adapter can be visualised and monitored in the Diffusion management console, using the `Network` tab.

Configuring the CDC adapter

Use the `configuration.json` file or environment variables to configure how the adapter shares data between Diffusion and databases.

The adapter can be started with only bootstrap configuration, with only a configuration file or with a combination of both. Bootstrap configuration refers to the details required to connect to a Diffusion server, which are server URL, principal and password. These bootstrap configurations can be passed as system properties or environment variables, which will be used by default, if they are not set in the configuration file.

The allowed system/env properties are:

```
diffusion.gateway.server.url
diffusion.gateway.principal
diffusion.gateway.password
gateway.client.id
```

When the adapter is started, it registers itself with the server using its id (defined in the configuration file) and configuration. The configuration will be empty if a configuration file is not passed as an argument when starting the adapter.

The server persists this information in its local persistence file.

If a service is added/updated/removed, the configuration of the adapter changes during runtime and is updated in the server for the adapter ID. Hence, if the adapter is shut down and started again, there is no need to pass the configuration file again, as the adapter uses the persisted configuration from the server.

To override the configuration persisted in the server, you can pass a configuration file as an argument. The passed configuration will replace the configuration persisted in the server and will be used to set up expected services.

Mapping to Diffusion Topics

Each event (insert/update/delete/read) from the database is captured and published to the server.

The adapter supports four different ways to map these events to a Diffusion topic.

1. Object : A table is mapped to a JSON topic, with each record being a JSON object keyed by the table's primary key. If a table does not have a primary key, updates for this table are ignored.

2. Array : A table is mapped to a JSON topic, with all records as entries in an array. If a table does not have a primary key, updates for this table are ignored. For 'Create' event, new entries are added at the end of the JSON topic's value. For 'Delete' event, the corresponding item in the value of the JSON Diffusion topic is updated to null.
3. Row : Each table row are mapped to individual JSON topic. The topic is created using database name, table name and primary keys. E.g: database/table/pk1,pk2. If a table does not have a primary key, and this setting is used, updates for this table are ignored.
4. None : As per Row, but the topic contents are exactly as received from Debezium. This will include schema information, as well as the table row data before and after the change.

> **Note:** If a table has composite primary key, the values of those keys will be escaped and concatenated together with ',' to formulate a complete primary key combination, which will be used in Object and Row topic mapping, as defined above.

The publisher service supports the following two operations, which can be invoked via the Diffusion console:

1. Pause - will close Debezium engine for the service
2. Resume - will restart Debezium engine for the service

Running the CDC adapter

The CDC adapter is not enabled by default.

Starting the adapter

The adapter requires a Java installation. The Java requirements are the same as for the Diffusion server.

Run the JAR file provided as cdc-adapter-6.7.7.jar.

There are several ways to run the adapter:

1. By passing the configuration file as an argument. The passed configuration will be used to override the persisted configuration in the server and set up required services.

   ```
   java -jar cdc-adapter-6.7.7 configuration.json
   ```
2. By only passing bootstrap configurations as system properties. If the server contains configuration for this adapter id, it will be used to initialise the adapter.

   ```
   java -jar -Dgateway.client.id=testCdcAdapter -
   Ddiffusion.gateway.server.url=ws://localhost:8080
   -Ddiffusion.gateway.principal=admin -
   Ddiffusion.gateway.password=password cdc-adapter-6.7.7.jar
   ```
3. By passing bootstrap configurations as system properties and configuration file.

   The passed configuration will be used to override the persisted configuration in the server and set up required services. If there is bootstrap configuration defined in the configuration file, it will override any passed as system properties.

   ```
   java -jar -Dgateway.client.id=testCdcAdapter -
   Ddiffusion.gateway.server.url=ws://localhost:8080
   -Ddiffusion.gateway.principal=admin -
   Ddiffusion.gateway.password=password cdc-adapter-6.7.7.jar
   configuration.json
   ```

> **Note:** To use the adapter with a database other than MySQL and PostgreSQL, the Debezium connector JAR for the database you are using needs to be added to the classpath when running the adapter.

Set up a secure connection with Diffusion

1. Import the Certificate Authority certificate which is used to sign the server's certificate to the local truststore:

   ```
   keytool -keystore local.client.truststore.jks -alias <alias> -import
   -file <ca-cert file name> -storepass <password> -keypass <password>-
   noprompt
   ```
2. Add the following system properties to the Java run command:

   ```
   -Djavax.net.ssl.trustStore=./local.client.trustStore.jks -
   Djavax.net.ssl.trustStorePassword=<password>
   ```
3. Update the URL of the Diffusion server in application.conf to use "wss://"

Demos

Diffusion comes with demo applications that demonstrate certain features of Diffusion.

If you chose the option to deploy the demos as part of the installation process, you can access them through the Diffusion landing page in your browser (typically at https://localhost:8080).

The demo front ends are deployed to `/html/demos` within your Diffusion installation directory.

Source code for the demos is available at `/examples/demos` within your Diffusion installation directory.

To remove the demos, delete `/html/demos` and `/examples/demos`.

Demos

The demos in the **Demos** section of the default Diffusion installation web page demonstrate various applications of Diffusion.

To access the Diffusion demos, start the server and type in the following URL into a browser: https://localhost:8080. (You may need to modify this if your Diffusion server is on a different machine or has been set to use a different port.)

Note that the Sportsbook demo publishing client must be started by running:

```
java -jar sportsbook-demo-publisher-jar-with-dependencies.jar
```

from within `/examples/demos`.

Table 53: Demos provided with the Diffusion server

Drawing Board demo	Multiple users draw on a virtual blackboard with colored chalk. All clients are updated in real time with strokes from the other clients. Open this demo in multiple browsers to see the responsiveness of client interactions. Requires Canvas support in the browser.
Sportsbook demo	Demonstrates live, high-frequency updating of betting odds for a list of football matches. Includes a Java publishing client which provides

	a list of odds as a JSON topic, and a viewer which subscribes to the topic using in-browser JavaScript.

To view the source code of the demos, go to `/examples/demos` within your Diffusion installation folder.

Part VI

Upgrading Guide

If you are planning to move from an earlier version of Diffusion, review the following information about changes between versions.

We recommend that you upgrade to the latest version of Diffusion as soon as you can.

When upgrading across multiple versions, ensure that you review the release notes and upgrade steps for all intermediate versions. For example, if you are upgrading from version 6.2 to version 6.4, first review the upgrade steps from version 6.2 to 6.3, then follow the steps to upgrade from version 6.3 to 6.4.

Release notes including known issues are available at the following location: http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html

For more information about Diffusion versions and support and upgrade policy, see the Support Center.

Related concepts

What's new in Diffusion 6.7? on page 23
The latest version of Diffusion contains new features, performance enhancements and bug fixes.

In this section:

- Interoperability
- Upgrading from version 5.x to version 6.0
- Upgrading from version 6.0 to version 6.1
- Upgrading from version 6.1 to version 6.2
- Upgrading from version 6.2 to version 6.3
- Upgrading from version 6.3 to version 6.4
- Upgrading from version 6.4 to version 6.5
- Upgrading from version 6.5 to version 6.6
- Upgrading from version 6.6 to version 6.7
- Upgrading to a new patch release

Interoperability

If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Interoperation between clients and servers

The following table describes which API client versions interoperate with which server versions.

The general rule is that an older client can connect to a newer server, but a newer client cannot connect to an older server.

Table 54: API interoperation

	Server version			
Client version	**6.4**	**6.5**	**6.6**	**6.7**
6.4	✓	✓	✓	✓
6.5	✗	✓	✓	✓
6.6	✗	✗	✓	✓
6.7	✗	✗	✗	✓

Interoperation between servers

Replication

All Diffusion servers within a cluster must be at the same minor version level.

For example, a cluster can include a mixture of 6.4.2 and 6.4.5 servers, but a cluster with a mixture of 6.4 servers and 6.7 servers is not supported.

Server versions	**6.4**	**6.5**	**6.6**	**6.7**
6.4	✓	✗	✗	✗
6.5	✗	✓	✗	✗
6.6	✗	✗	✓	✗
6.7	✗	✗	✗	✓

Remote topic views

Remote topic views are supported between servers of version 6.5.0 or above.

Fan out

Fan-out distribution is supported between servers of different versions.

Server versions	6.4	6.5	6.6	6.7
6.4	✓	✓	✓	✓
6.5	✓	✓	✓	✓
6.6	✓	✓	✓	✓
6.7	✓	✓	✓	✓

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Upgrading from version 5.x to version 6.0

Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as publishers and authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

> **Note:**
>
> The Classic API has been removed and is no longer supported. To upgrade your solution to use Diffusion version 6, you must create clients that use the Unified API.
>
> From this point onward all references to clients are to Unified API clients.

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

The minimum supported framework or platform version for a number of our client libraries has changed:

- Java client must use Java 8 or above (minimum update 1.8.0_131-b11).

The C client now depends on zLib. Ensure that you have zLib version 1.2 or later available on your development system and add `-lz` to the LDFLAGS in your Makefile. For more information, see http://www.zlib.net.

Some features that your client applications might use have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in following locations:

- The release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html
- The API documentation located at http://docs.pushtechnology.com/docs/6.0

The following table lists API classes and methods that have been removed. If you attempt to recompile application code that uses these classes or methods against the version 6.0 APIs, it fails. Rewrite your application code to not include these features.

Table 55: API features removed in version 6.0

API affected	Removed feature	Suggested alternative
Classic API	Every Classic client API	Rewrite your clients to use the API
All APIs	Methods and classes related to removed topic types, see Removed components on page 652.	Use alternative topic types.

API affected	Removed feature	Suggested alternative
All APIs	Methods and classes related to removed transport protocols, see Removed components on page 652.	Use alternative transport protocols.
Apple API	All methods that use `content` when creating and interacting with topics and messages.	Equivalent methods that use `value`.
Apple API	`PTDiffusionSessionErrorHandler`	The `error` property on `PTDiffusionSessionState`
Apple API	`diffusionTopicStream:didUnsubscribeFromTopicPath:reason:` in `PTDiffusionTopicStreamDelegate`	`diffusionTopicStream:didUnsubscribeFromTopicPath:reason:`
Apple API	`removeTopicStream:` in `PTDiffusionTopicsFeature`	`removeStream:`
.NET API	The `SetSessionDetailsListener`, `SetSessionDetails`, and `GetSessionDetails` in `IClientControl`	Use a session properties listener instead.
.NET API	`Session.Start` method and the CONNECTED_INITIALISING and RECOVERING_FAILOVER states	None
Java API	The `setSessionDetailsListener`, `setSessionDetails`, and `getSessionDetails` in `ClientControl`	Use a session properties listener instead.
Java API	`ClientControl.close` methods that include a String `reason` parameter	Use `ClientControl.close` methods that do not include this parameter. The `reason` parameter is not passed to the client being closed. If you want to notify the client being closed of the reason for its closure, use the `MessagingControl` feature to send a message to the client. To ensure that the message is received before closing the client session, wait for callback to return before calling `ClientControl.close`.
Java API	`Topics.removeTopicStream` method	`Topics.removeStream` method
Java API	`TopicControl.removeTopics` method	`TopicControl.remove` method
Java API	`sendFetchReply` method	None
Java API	`Session.start` method and the CONNECTED_INITIALISING and RECOVERING_FAILOVER states	None
Java API	`SessionClosedException` exception thrown from synchronous methods.	

The following table lists API classes and methods that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 56: API features deprecated in version 6.0

API affected	Deprecated feature	Suggested alternative
Apple API	`PTDiffusionTopicDetails`	`PTDiffusionTopicSpecification`
Apple API	`sendWithTopicPath`	`sendWithPath`
Java API	`Messaging.getStreamsForTopic`	None
Java API	`TopicDetails`	`TopicSpecification`
.NET API	`IMessaging.GetStreamsForTopic`	None
.NET API	`ITopicDetails`	`ITopicSpecification`
JavaScript API	`TopicDetails`	`TopicSpecification`
All APIs	Setting headers on one-way messages	Send information in the message body instead. Note this only applies to the legacy one-way messaging. Custom headers are not supported for request-response messaging.
Publisher API	Unused values in `TopicProperty`: `LOAD_ENCODING`, `DELTA_ENCODING`, `LOAD_HEADERS`, `LOAD_ACK_REQUIRED`, `DELTA_ACK_REQUIRED`. Methods in `PublishingTopicData` that get and set these values.	None
Configuration API	`ServerStatisticsConfig`	None

The following list includes behavior that has changed in the API. If your application code relies on the previous behavior, rewrite your application code to take into account the new behavior.

- Java API methods throw a `NullPointerException` when a null value is provided for a non-optional parameter. In previous releases, an `IllegalArgumentException` was thrown.
- Slave topics created by the client API no longer require that the master topic exists when the slave topic is created. The behavior of slave topics created by the Publisher API remains unchanged.
- Slave topics created by the client API are no longer removed when the master topics they are linked to are deleted. The behavior of slave topics created by the Publisher API remains unchanged.
- If a client session selects a set of topics to remove, all those that the session has permission to remove are removed. This is a change from the previous behavior, where the remove operation failed if the session did not have permission for one or more of the selected topics.
- "Remove with session" now removes only those topics that the session has permission to remove at the time the session closes. Previously, "remove with session" was able to remove all topics in a branch if the session had remove permission at the branch path.
- The client API can no longer remove topics created using the Publisher API and the Publisher API can no longer remove topics created using the client API.
- The Publisher API no longer supports message filters. It is no longer possible to filter topic updates on a session by session basis. Session properties filters can be used to manage subscriptions to topics and to address messages sent to sessions.

- The Publisher API no longer supports message acknowledgements. The Publisher API methods and configuration that control acknowledgements now have no effect and are deprecated.
- Topics created using `TopicDetails` (in Java/JavaScript), `ITopicDetails` (in .NET) or `PTDiffusionTopicDetails` (in Apple APIs) will not work with topic persistence. Use the equivalent `TopicSpecification` class instead.

Removed components

The following components have been removed from Diffusion in version 6.0:

- Legacy topic types:
  - Paged string topics
  - Paged record topics
  - Protocol buffer topics
  - Service topics
  - Child list topics
  - Topic notify topics
  - Custom topics
- Legacy transport protocols:
  - DPT
  - HTTP Duplex
  - HTTP Chunked Streaming
- The Introspector/Eclipse plugin

  Instead use the console.
- JMS adapter version 5.1

  Instead use the latest JMS adapter.

Upgrading your server installation

> **Note:**
>
> At release 6.0, the Diffusion server is tested and supported on Oracle® JDK 8 (minimum update 1.8.0_131-b11).
>
> Earlier versions of Java are not supported.

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation. When you do, consider making the following changes:

   - The following configuration items are no longer valid. Remove them from your configuration files.

     In `Connectors.xml`

     `acceptors` element

     In `Logs.xml`

     `rotate-daily` element

     In `Management.xml`

     JMS user and password definitions

In `Publishers.xml`

`servers` element

In `Servers.xml`

`multiplexers` and its child elements, `write-selectors`, `auto-fragment`, `retry-interval` in the fan-out configuration, elements that set thread priority, and outbound thread pool configuration.

In `WebServer.xml`

`websocket-secure-response` and `close-callback-requests`

- The following configuration items are now deprecated and are ignored. Consider removing them from your configuration files.

  In `Connectors.xml`

  `type` and `policy-file` elements in the `connectors` definition.

  In `Publishers.xml`

  `topic-aliasing` element

Behavior changes at the Diffusion server

The following list includes behavior that has changed at the server. If your solution relies on the previous behavior, adjust your solution to take into account the new behavior.

- Topic replication is no longer supported for these deprecated topic types:
  - single value
  - record

  Replace single value topics with the new explicitly-typed topics. Replace record topics with the new recordV2 topic type.
- Topic replication is no longer supported for publisher-created topics.
- The `Connectors.xml` file included in the installation no longer includes `backlog` elements to define the maximum number of incoming clients that can be waiting to connect. Instead the Diffusion default of 1000 is used. For more information, see Connectors on page 502.
- The `Statistics.xml` `server-statistics` element and the related `ServerStatisticsConfig` API have no function and are deprecated. The `Statistics.xml` file included in the installation no longer includes the `server-statistics` element.
- The `enabled` attribute on the replication element of the `Replication.xml` configuration file has been deprecated. The related methods for the `ReplicationConfig` interface have also been deprecated. The attribute has been made optional and defaults to true. The attribute continues to control whether replication is enabled. Replication can be controlled by the individual settings for session and topic replication.
- The JMS adapter has moved from the `adapters` directory in the Diffusion server installation to the `adapters/jms` directory.
- The Push Notifications bridge has moved from the `pushnotifications` directory in the Diffusion server installation to the `adapters/pushnotifications` directory.

Related concepts

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661

Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.0 to version 6.1

Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as publishers and authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

The .NET client library is now fully compatible with the .NET Standard 2.0. Officially supported platforms are Windows (via .NET Framework, .NET Core), MacOS (.NET Core), and various Linux distributions (via .NET Core). In order to make this change

clear we removed the old `PushTechnology.ClientInterface` library and NuGet package and replaced it with `Diffusion.Client`. This requires re-compilation of your project.

In the Javascript client, zlib code for message decompression has been separated out into `browserify-zlib-0.2.0.js`. Include `browserify-zlib-0.2.0.js` for clients that want to make use of the client compression capability. This can be achieved at build time by using browserify to package the browserify-zlib npm module into the application library.

`browserify-zlib-0.2.0.js` is included in the `clients/js` directory of your Diffusion installation, and can be downloaded from JavaScript SDK downloads.

Clients will log out a warning at startup if `browserify-zlib-0.2.0.js` is not included. The client's initial connection request will set the per-message compression capability depending on whether `browserify-zlib-0.2.0.js` is included. This will indicate to the server whether messages should be compressed before they are sent to the client.

The Java client now supports Java 9, 10 and 11. Java 8 is still required for the server.

Some features that your client applications might use have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in following locations:

- The release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html
- The API documentation located at http://docs.pushtechnology.com/docs/6.1

The following table lists API classes and methods that have been removed. If you attempt to recompile application code that uses these classes or methods against the version 6.1 APIs, it fails. Rewrite your application code to not include these features.

Table 57: API features removed in version 6.1

API affected	Removed feature	Suggested alternative
.NET API	ISession.GetXXXFeature() methods	ISession.GetXXXFeature() methods Use the equivalent ISession.XXX property instead. For example, instead of ISession.GetClientControlFeature() use ISession.ClientControl
.NET API	SessionId class	Use ISessionId property instead.
.NET API	DefaultStreamCallback and StreamDefault	Use DefaultStream instead.
.NET API	ISubscriptionRequest	
.NET API	IClientSession	
.NET API	IClientControlFeatureHandler	
.NET API	IQueueListener	
.NET API	IStateListener	
.NET API	IAuthControlFeatureHandler	

API affected	Removed feature	Suggested alternative
.NET API	ITopicManagementHandler & ITopicManagementFactory	
.NET API	IClientInfo	
.NET API	IPingDetails.RoundTripTime property	

The following table lists API classes and methods that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 58: API features deprecated in version 6.1

API affected	Deprecated feature	Suggested alternative
All	removeTopicsWithSession	Use the new REMOVAL topic property.
	TopicEventListener (subscriber notifications) methods in the TopicControl feature	These methods have been deprecated as they only worked for local sessions. The use case for these methods was typically removal of unused topics which is now better addressed using the REMOVAL topic property (which is cluster aware).
	Content interface	The Content interface is being phased out. It is still used in some interfaces, such as Fetch and Messaging when receiving data. In all other cases its use should be avoided as it will be removed at a future release. At this release some remaining interfaces that used Content (such as builders and readers) have been deprecated.
.NET API	`IServiceRequest`	
.NET API	`ISessionFactory.SslContext(RemoteCertificateValidationCallback, LocalCertificateSelectionCallback)`	Use the `ISessionFactory.SslContext(RemoteCertificateValidationCa` method instead.
.NET API	`IClientEndpoint`	None
Publisher API	See release notes for details	

Removed components

The following components have been removed from Diffusion in version 6.0:

- Cascade URLs for reconnection strategies have been removed from the C client API. Applications will no longer be able to supply a list of server URLs to connect to in the scenario that a C client session loses its connection a server. Reconnection is now targeted only at the server to which the connection was lost.

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation. When you do, consider making the following changes:

   - The `server-statistics` elements in `Statistics.xml` and the `ServerStatisticsConfig` are no longer valid and should be removed.
   - The customConfigurator element in `Replication.xml` configuration is deprecated and is ignored by the server. This was intended to enable Kubernetes support but was never documented. Consider removing it from your `Replication.xml`. (Use <replication kubernetes-enabled="true"/> to enable Kubernetes support if it is included in your licence.)

Behavior changes at the Diffusion server

The following list includes behavior that has changed at the server. If your solution relies on the previous behavior, adjust your solution to take into account the new behavior.

1. Fan-out connections no longer appear in session property queries and session property listener notifications. Previously fan-out connections would appear in session property queries and would also be notified to session property listeners as if they were client connections. This is no longer the case.
2. If client statistics are enabled in `Statistics.xml`, a summary of the connected sessions is regularly logged. In previous releases, these reports were logged to a separate log file, specified by the log-file configuration property. From this release, the reports are logged to the server log. The log-file configuration property now has no purpose, and has been deprecated together with the corresponding configuration API property. If separate log files are required or the reports are not desired, use a third-party SLF4J logging back-end such as Log4j 2, and configure it appropriately to partition or filter the server log.
3. The server no longer maintains rates for statistic metrics. Previously, the server maintained exponentially-weighted moving averages for some metric values, calculated over 1 minute, 5 minute and 15 minute intervals. These were available through the JMX statistics API and (if specially configured) the Diffusion console. The derived rates were rarely used and have been removed to reduce the CPU and memory overhead of statistics.
4. Some rarely used statistics have been remove to improve performance. These statistics were only available through JMX and include the connection count per connection type; the number of publishers started and loaded; the publisher name; the per-publisher subscription count; the global count of subscriptions; the per-session connection type; and the per-topic counts of subscriptions added and removed.
5. At previous releases, if a server was created in a programmatic environment (without XML configuration files) then it was necessary to set up certain configuration items in order to establish a working server that could accept normal web clients. All configuration items now have sensible defaults, enabling a working server to be started without any specific configuration.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664

Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.1 to version 6.2

Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as publishers and authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

The Java client now supports Java 9, 10 and 11. Java 8 is still required for the server.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in following locations:

- The release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html
- The API documentation located at http://docs.pushtechnology.com/docs/6.2

The following table lists API features that have been removed. If you attempt to recompile application code that uses these classes or methods against the version 6.2 APIs, it fails. Rewrite your application code to not include these features.

Table 59: API features removed in version 6.2

API affected	Removed feature	Suggested alternative
All	Record topics	Rewrite your application to use either JSON or recordV2 topics
All	Single value topics	Use string, int64 or double topics as appropriate
All	Stateless topics	Use an appropriate topic type with the DONT_RETAIN_VALUE topic property enabled
All	addTopic methods that use TopicDetails	Use methods that take TopicSpecification
All	addTopic methods that take an initial value	It is no longer possible to add a topic with an initial value. However, the new Update feature allows for dynamic creation of topics that do not exist, which achieves the same effect.
All	Creation of routing topics with server side handlers	Applications should be changed to provide client side routing topic handlers.
All	Other items deprecated at 6.0	Consult 6.0 API documentation for alternatives
Publisher	Client send methods	These only worked with stateless topics and TopicStreams, which have both been removed
Publisher	Other items deprecated at 6.0	Consult 6.0 Publisher API for alternatives
.NET API	IClientInfo	
.NET API	IPingDetails.RoundTripTime property	

The following table lists API classes and methods that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 60: API features deprecated in version 6.2

API affected	Deprecated feature	Suggested alternative
All	TopicUpdateControl	Use the new TopicUpdate feature.
All	Topics.feath methods	Use the new fetchRequest

API affected	Deprecated feature	Suggested alternative
All	One-way messaging	Use request-response messaging instead
All	TopicDetails, RoutingTopicDetails, SlaveTopicDetails	Only retained to support the deprecated getTopicDetails method
All	Content	Only used in deprecated features
All	SessionDetails, ClientSummary, ClientLocation	Only used by deprecated Authentication Handler interfaces
All	SendOptions and ReceiveContext	Only used in deprecated one-way messaging methods
All	TopicAddFailReason.USER_CODE_ERROR	No longer used
.NET, Java, Android and C Clients and Publisher	AuthenticationHandler	Use the new Authenticator interface
Publisher	A number of Publisher API interfaces and methods are deprecated; see release notes	Consult the Publisher API documentation for guidance

Change of behavior

All of the Publisher and Topic `addTopic` methods, other than those that take a `TopicData` parameter have now been deprecated. For backwards compatibility, those methods that did not specify the type of `TopicData` (which would have previously created a topic of the stateless topic type) will now create a binary topic with its DONT_RETAIN_VALUE property set to true.

Also, where a topic is created that leads to the creation of intermediate topics, then intermediate binary topics will also be created. This differs from topics created by the client API where intermediate nodes would be unbound (that is, have no topic).

For reasons of backwards compatibility with other Publisher API interfaces, it is not possible to create unbound nodes using the Publisher API. For maximum efficiency, use the client API to create topics instead of the Publisher API.

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

   The following configuration items are no longer valid. Remove them from your configuration files:

   - `Connectors.xml`: remove connector.type and connector.policy-file elements
   - `Publishers.xml`: remove publisher.topics, publisher.topic-aliasing, publisher.ack-timeout, publisher.auto-ack elements
   - `Replication.xml`: remove enabled attribute and topics.topicPath element

- `Server.xml`: remove log-message-data and conflation elements
- `WebServer.xml`: remove comet-bytes-before-new-poll and comet-initial-message-padding elements

The following configuration items are now deprecated. Consider removing them from your configuration files.

- `ConnectionValidationPolicy.xml`: remove policy.automatic attribute
- `Server.xml`: default-load-message-capacity and default-delta-message-capacity elements.
- `Statistics.xml`: publisher-statistics element and enabled, client-statistics.enabled and topic-statistics.enabled attributes.

Behavior changes at the Diffusion server

The topic tree is now sorted into lexical path name order. Previously topics were stored in the topic tree in creation order within parent node. If your application relies on topics being ordered by creation time, you may need to adjust your application.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.2 to version 6.3

Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as publishers and authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

The Java client now supports Java 9, 10 and 11. Java 8 is still required for the server.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in following locations:

- The release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html
- The API documentation located at http://docs.pushtechnology.com/docs/6.2

The following table lists API classes and methods that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 61: API features deprecated in version 6.3

API affected	Deprecated feature	Suggested alternative
Publisher API	PublisherStatistics	Use clients instead of publishers.
Publisher API	ClientStatistics	Use JMX/Prometheus/console, metric collectors
Publisher API	TopicStatistics	Use JMX/Prometheus/console, metric collectors

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.

3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

   The following configuration items are now deprecated. Consider removing them from your configuration files.

   - `Statistics.xml`: topic-statistics, publisher-statistics, reporters and server-statistics elements.
4. The working copies of the `Security.store` and `SystemAuthentication.store` files are now stored in the `persistence` directory. Copy those files from the `etc` directory of your previous installation to the `persistence` directory of your new installation.
5. The `hazelcast.xml` configuration file is now stored in the `etc` directory rather than the `data` directory. If you have customised `hazelcast.xml`, copy it from the `data` directory of your previous installation into `etc` in the new installation.

Behavior changes at the Diffusion server

Topic views, metric collectors and the security/system authentication stores are now replicated across a Diffusion cluster if any form of replication is enabled.

If you are using clustered servers, do not edit the security/sytem authentication store files directly. Update the content using a client.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647

If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.3 to version 6.4

Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

The server now supports Java 11 as well as Java 8. See System requirements for the Diffusion server on page 27 for further details.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in the release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html

The following table lists features that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 62: API features deprecated in version 6.4

API affected	Deprecated feature	Suggested alternative
Most clients	MissingTopicNotification.cancel	Use the new fetch API; the legacy fetch mechanism is deprecated.
.NET	TopicSelectors and methods that accept them	Use strings where a TopicSelector is required as an argument.

API affected	Deprecated feature	Suggested alternative
Publisher API	Publishers and the Publisher API are now deprecated	Use clients instead.
Logging	The built-in logging implementation, referred to as "default logging framework" in earlier versions and configured with etc/Logs.xml, is no longer the default.	Log4j2
Slave topics	Slave topics are now deprecated.	Use topic views instead.

The following table lists API classes and methods that have been removed. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 63: API features removed in version 6.4

API affected	Removed feature	Suggested alternative
.NET	Topics.GetTopicDetails removed	
.NET	MissingTopicNotification.cancel removed	
Server	MBean setQueueDefinition()	

Change of behavior

The installer has been rewritten for this release. The new installer is based on the third party izpack framework. The installation options have been simplified. In previous releases, both the installer and a separate product distribution jar file were required. In this release, there is a single installer file that should be run using Java: `java -jar diffusion-installer-6.4.0.jar`

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

   The following configuration items are now deprecated. Consider removing them from your configuration files.

   - `Logs.xml`: Log4J2 is now the default logging implementation; all Logs.xml settings but default log directory and console monitored log are ignored on a default install.

Behavior changes at the Diffusion server

In previous releases, automatic topic removal (via the REMOVAL topic property) did not take into account sessions (or subscribers) at secondary (or downstream) fan-out servers and was therefore of limited use in such situations. From this release all sessions and subscriptions at downstream fan-out servers are taken into account when evaluation topic removal conditions.

The COMPRESSION topic property now accepts the values "off", "low", "medium" and "high", and the default if no level is set is now "low". The previous values were "false" (equivalent to "off") and "true" (equivalent to "medium") with "true" as the default. As a result, for topics where no COMPRESSION value has been set, the compression level used will be reduced from medium to low when upgrading to 6.4.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.4 to version 6.5

Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Server-side publishers written in Java are no longer supported. If your application uses publishers, replace them with clients. DAR files are no longer supported.

The server now supports Java 11 as well as Java 8. See System requirements for the Diffusion server on page 27 for further details.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in the release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html

The following table lists features that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 64: API features deprecated in version 6.5

API affected	Deprecated feature	Suggested alternative
All	"TopicPermission" methods have been deprecated because the name is misleading. Some of the permissions they cover apply to message paths or to session lock names, which are separate from topics.	Use new PathPermission methods (equivalent apart from the name).
All	The MessagingControl feature has been deprecated and all types and methods have been moved to the Messaging feature.	Use types and methods in Messaging for compatibility with legacy code.
All	The ClientServiceConfig compressionThreshold setting is no longer used.	The efficiency of HTTP polling transport when the client supports 'deflate' compression has been improved, so this setting is ignored and will be removed in a future release.

The following table lists API classes and methods that have been removed. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 65: API features removed in version 6.5

API affected	Removed feature	Suggested alternative
All	One-way messaging	Use request-response messaging instead

API affected	Removed feature	Suggested alternative
All	removeTopicsWithSession	Use REMOVAL topic property instead
Publisher API	Most publisher functionality removed as publishers no longer supported by server	Replace publishers with clients
C Client	`remove_topics` function	Use `topic_removal` instead
JMS Adapter one-way messaging support	The JMS Adapter no longer uses Diffusion one-way messaging.	The adapter now supports request-response messaging instead.

Connection security configuration changes

Diffusion 6.5 now supports secure connection to the server with TLSv1.3 (in addition to earlier TLS/SSL protocols). If you wish to use TLSv1.3, ensure that the Java runtime you are using supports it. It is available in most Java 11 JDKs, but is less well supported by Java 8 JDKs.

The TLS versions used by the Diffusion server and Diffusion Java client are no longer configured with the diffusion.tls.protocols system property. To enable or disable particular protocols, use the interfaces provided by the JDK. For the Oracle JDK, see the Oracle JDK cryptographic configuration documentation.

The cipher suites used by the Diffusion server and Diffusion Java client are no longer configured with the https.cipherSuites system property, which is now ignored. To enable or disable particular ciphers, use the interfaces provided by the JDK.

See Network security on page 487 for more information.

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

   The following configuration items are now deprecated. Consider removing them from your configuration files.

   - `Publishers.xml`

Path permission evaluation with multiple roles

This release significantly changes the evaluation of path permissions for a client session with multiple roles.

In previous releases, path permission rules from different roles were merged before evaluation, so creating a permission rule for a particular path would mask rules for parent paths from other roles.

From this release, path permissions are evaluated independently for each role. A session has a permission if any of its assigned roles has that permission.

A new `isolate path` security language statement has been added that disables inheritance of path permissions assigned to parent paths, allowing a branch of the path hierarchy to be configured independently.

If a Diffusion 6.5 server is started against a security store file from Diffusion 6.4 or earlier, the file will automatically be translated to an equivalent configuration under the new rules. For each path in a path permission assignment for a role, a separate statement is added to isolate the path.

In practice you will typically find that many of the path isolation statements can be removed without affecting your application's security policy, resulting in a simpler configuration.

See Permissions on page 142 for more details about how the new system works.

New management console

The web-based management console has been completely redesigned, with many improvements.

The classic console, as found in Diffusion 6.4, is still available, but will be removed in a future release.

If you select the console during installation, both the new and classic consoles will be installed.

Behavior changes at the Diffusion server

- In previous releases, changes to the security store which altered read_topic permission assignments were not applied to existing subscriptions until the role assigned to an existing session changed (if the session reauthenticated, or a control session changed the session's role).

  From this release, changes to the security store which affect read_topic permissions are immediately applied to all sessions, resulting in the appropriate subscriptions and unsubscriptions. Each session's topic selections will be re-evaluated against topics for which the session has gained read_topic permission, and sessions will be unsubscribed from topics for which they no longer have read_topic permission.
- Performance with large numbers of path permissions has been greatly improved, enabling scaling up to millions of path permissions.
- From this release, the session ID and security principal are included as additional fields in log messages where they are available.

  The log fields are available in the Log4j thread context map under the `session` and `principal` keys. You can use this for context-sensitive logging: for example, logging debug messages only for a particular security principal. You can configure this without restarting the server by setting a new log configuration using JMX.

  If you are upgrading an existing configuration, review `etc/log4j2.xml` from a fresh installation to see the required changes. Also, see `etc/log4j.xml` for a commented example of context-sensitive logging.
- Server-side publishers are no longer supported. The Publisher API is now known as the Server API. Publisher-related functions have been removed. The remaining API is used for server configuration and running Diffusion embedded within a Java application. See Server API documentation for details.
- The management console is no longer packaged as a DAR file. The installer deploys the console (and classic console) within `/html/`.
- The demos are no longer packaged as DAR files. If you choose to install demos and examples, the installer copies the source files for the demos into `/examples/demos`. If you also select the option to deploy the demos, they will be deployed in `/html/demos`.
- Path permission evaluation with multiple roles has changed, but your existing configuration will be automatically updated to keep the same behavior (see above). You should check that the updated configuration behaves correctly.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.5 to version 6.6

Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

API changes

Further information about removed or deprecated features is available in the release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html

The following table lists features that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 66: API features deprecated in version 6.6

API affected	Deprecated feature	Suggested alternative
.NET	IStream methods of the SendRequestToFilter IFilteredRequestCallback	Use the task returned by SendRequestToFilterAsync to detect errors.
JavaScript Client	onError and onClose methods of the sendRequestToFilter FilteredResponseHandler	Use the Result returned by sendRequestToFilter to detect errors.
Java & Android Clients	SERVER_CLOSING CloseReason in ClientControl	This close reason is never sent by the server.

The following table lists API classes and methods that have been removed. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 67: API features removed in version 6.6

API affected	Removed feature	Suggested alternative
All	AuthenticationHandler interface	Use Authenticator instead.
All	TopicEventListener and TopicEventStream interfaces	Use REMOVAL topic property to remove topics instead.
All	TopicUpdateControl interface	Use TopicUpdate instead.
All	Legacy fetch API	Use the updated fetch API (introduced in version 6.2) instead.
All	Slave topics	Use topic views instead.
Push Notification Bridge	The push notification bridge has been removed.	The push notification adapter packaged in previous versions can be used against this release. Please contact Push Technology if you are using the adapter, or need to integrate Diffusion with push notification servers.

API affected	Removed feature	Suggested alternative
ClientService.Config compressionThreshold setting	This setting has been removed.	The efficiency of HTTP polling transport when the client supports 'deflate' compression has been improved so this setting has been ignored since version 6.5.
All	session_set_maximum_outbound_queue_size	Use diffusion_session_factory_maximum_queue_size instead.
C client	Session wills	Use automatic topic removal instead.

Hazelcast version change

The version of the built-in Hazelcast datagrid used has been updated from Hazelcast version 3.12 (in Diffusion 6.5) to Hazelcast IMDG version 4.02.

A default configuration file compatible with version 4.02 is provided at `etc/hazelcast.xml`.

If you have customized your Hazelcast configuration file, you may need to update it to work with 4.02 after copying it into the new installation.

See the Hazelcast™ Reference Manual for more information.

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.
4. The configuration format for the JMS adapter has changed, and it no longer supports request-response messaging. You will need to convert your existing XML configuration into the new JSON format. See Configuring the JMS adapter on page 640 for details.

Behavior changes at the Diffusion server

Slave topics are no longer supported by the server. Older clients will no longer be able to create slave topics. Slave topics stored in persistence files will not be restored.

The server will no longer accept connections from Diffusion versions before 6.2.

Support for the `SessionDetails`-based API features available via version 5.9 clients has now been removed. Release 5.9 clients will no longer be able to use the `getSessionDetails` and `SessionDetailsListener` capabilities with a version 6.6 server.

Support for the `getTopicDetails` service was removed from clients in version 6.4, and has now been removed from the server.

The default license file name is now `license.lic`. If `license.lic` is not found, `licence.lic` will be used if it is present.

Missing topic notification handling has been simplified. Previously, subscription requests would not be processed until the server received a response from the missing topic handler. From this release, subscription requests are processed immediately by the server, without waiting for the handler.

- The proceed() and cancel() methods of `MissingTopicNotification` no longer have any effect and are deprecated.
- A missing topic handler can no longer cancel a subscription. A supported alternative is to use the `SubscriptionControl` feature to issue a compensating unsubscribe operation.

Topic views now respect the OWNER property when evaluating source topics. As a result, a source topic is now selected for evaluation by a topic view if the principal of the topic view is the same as the owner of the source topic.

- The OWNER property (if present) is now copied from a source topic to all reference topics that are derived from it.
- Granting ownership of a source topic also grants ownership of all derived reference topics.
- Both the topic view principal and the OWNER property (if set) are now treated as owners of a reference topic. This means that the principal of the view which created a reference topic has full access to it, as well as any OWNER principal specified.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading to a new patch release on page 676
When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading from version 6.6 to version 6.7

Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Upgrading your applications

Server-side components

Recompile all Java application components that are deployed to the Diffusion server, such as authorization handlers, against the new version `diffusion.jar` file. This file is located in the `lib` directory of your new Diffusion server installation.

Some features that your Java application components might use have been removed or deprecated. Pay attention to new deprecation warnings and compilation failures that occur during recompilation and review the API changes information in the following section to see if these changes affect your applications.

Clients

You can choose not to recompile your client applications and continue to use client libraries from a previous release. If you choose to use client libraries from a previous release, ensure that the libraries are compatible with the new server. For more information, see Interoperability on page 647.

You can choose to upgrade your client applications to use the new client libraries. To do this, recompile the client applications against the client libraries located in the `clients` directory of your new Diffusion server installation and repackage your client application with the new library.

Your client applications may use features that have been removed or deprecated. Review the API changes information in the following section to see if these changes affect your applications.

The minimum Xcode version for the Apple SDK is now 12.4.

Note that the local install method for the Python SDK has changed. See the Client basics page for Python for details.

API changes

Further information about removed or deprecated features is available in the release notes provided online at http://docs.pushtechnology.com/docs/6.7.7/ReleaseNotice.html

The following table lists features that have been deprecated. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 68: API features deprecated in version 6.7

API affected	Deprecated feature	Suggested alternative
All SDKs	Routing topics	Use session trees.
All SDKs	Fan-out	Use remote topic views.
Java & NET clients	All methods that use callbacks	Use CompletableFutures (for Java) or task based variants (for .NET).
Java client	ClientControl.sessionIdFromString	Diffusion.sessionIdFromString
Java client	ClientControl.setQueueEventHandler(QueueEventHandler)	setQueueEventHandler(QueueEventStream)
Java client	ClientControl.setSessionPropertiesListener(SessionPropertiesListener)	setSessionPropertiesListener(SessionPropertiesStream)
Java client	TopicControl.addMissingTopicHandler(String, MissingTopicHandler)	addMissingTopicHandler(String, MissingTopicNitificationStream)

API affected	Deprecated feature	Suggested alternative
Java client	TopicControl.newSpecification	Diffusion.newTopicSpecification

The following table lists API classes and methods that have been removed. If your application code uses these classes or methods, consider rewriting your application code to not include these features.

Table 69: API features removed in version 6.7

API affected	Removed feature	Suggested alternative
All SDKs	MessagingControl feature interface	All functionality is available through the Messaging feature.
Java, Android, C, JavaScript clients	Unused TopicAddFailReason values: INITIALISE_ERROR, TOPIC_NOT_FOUND, USER_CODE_ERROR	none

Upgrading your server installation

To upgrade your Diffusion server installation, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Contact Push Technology for an updated license file.
3. You can copy most of your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

Behavior changes at the Diffusion server

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading to a new patch release on page 676

When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Upgrading to a new patch release

When upgrading to a new patch release there are typically no changes to the configuration values or the APIs. All that is required is to copy your existing files from the old installation to the new installation.

To upgrade to a new patch release, complete the following steps:

1. Use the graphical or headless installer to install the new version of Diffusion.

   For more information, see Installing the Diffusion server on page 398.
2. Copy your existing license file from your previous installation to the `etc` directory of your new installation.
3. Copy your existing configuration files from the `etc` directory of your previous installation to the `etc` directory of your new installation.

Related concepts

Upgrading from version 5.x to version 6.0 on page 648
Consider the following information when upgrading from Diffusion version 5.x to version 6.0.

Upgrading from version 6.0 to version 6.1 on page 654
Consider the following information when upgrading from Diffusion version 6.0 to version 6.1.

Upgrading from version 6.1 to version 6.2 on page 658
Consider the following information when upgrading from Diffusion version 6.1 to version 6.2.

Upgrading from version 6.2 to version 6.3 on page 661
Consider the following information when upgrading from Diffusion version 6.2 to version 6.3.

Upgrading from version 6.3 to version 6.4 on page 664
Consider the following information when upgrading from Diffusion version 6.3 to version 6.4.

Upgrading from version 6.4 to version 6.5 on page 666
Consider the following information when upgrading from Diffusion version 6.4 to version 6.5.

Upgrading from version 6.5 to version 6.6 on page 670
Consider the following information when upgrading from Diffusion version 6.5 to version 6.6.

Upgrading from version 6.6 to version 6.7 on page 673
Consider the following information when upgrading from Diffusion version 6.6 to version 6.7.

Related reference

Interoperability on page 647
If you plan to use Diffusion servers and clients of different versions together, review the following information that summarizes support between versions.

Appendix

Appendices

The appendices contain reference information.

In this section:

- Document conventions
- Glossary
- Trademarks
- Copyright Notices

Appendix A

Document conventions

This user manual uses certain typographic conventions to distinguish between different types of information.

The following table describes how different types of information are represented typographically.

Table 70: Typographic conventions used in this manual

Convention	Usage
`Monospace`	Indicates the following items: • Source code • Class or method names • Command names • File paths • Information input by the user
Bold	Indicates the following items: • Interface element titles (buttons, menu items, field names) • Window or panel titles
Italic	Indicates the following items: • New terms – when appearing in the text • Variable values – when appearing in code or syntax examples
Greater than sign (>)	Indicates a menu item or sequence of menu items. For example, “Choose **File** > **Save**” means choose the **Save** item from the **File** menu.
Highlighting	Indicates an example value in descriptive text.

Appendix B

Glossary

The glossary contains key terms associated with Diffusion and their definitions.

In this section:

- A
- C
- D
- E
- F
- G
- H
- I
- J
- L
- M
- N
- P
- Q
- R
- S
- T
- U
- V
- W
- X

A

acknowledgment

A deprecated mechanism for the recipient of a message or update to inform the sender of that message or update that it was received. Acknowledgements were removed in Diffusion 6.0. Methods related to acknowledgements are still present in the Publisher API, but are deprecated.

ack

API

Application Programming Interface
A set of contracts that you can program against to interact with Diffusion.

The following APIs are available:

- Server API
- Client APIs

API

API

API

APR

Apache Portable Runtime

Apache Portable Runtime (APR)

APR

APR

ASCII

American Standard Code for Information Interchange

A character-encoding scheme that encodes 128 specified characters into 7-bit binary integers.

ASCII

ASCII

ASCII

C

callback

An object, specific to a single call, that is used to respond to a request.

CBOR

Concise Binary Object Representation

Concise Binary Object Representation (CBOR)

CBOR

CBOR

certify

Push Technology certifies specific hardware and software version for use with Diffusion. Certified versions have been fully functional tested and performance tested with the Diffusion server. In addition, Push Technology supports some hardware and software that has not been certified.

client

An entity that connects to the Diffusion server and subscribes to topics.

Typically a client is a user-written application communicating with the Diffusion server through a client API or the Diffusion protocol.

client library

A library that is included in a client application to enable interaction with the Diffusion server.

conflation

The merging or replacing of a queued update with a newer update to reduce network traffic. Conflation removes outdated information from the queue of content to be sent and either replaces the outdated information with the conflated information or appends the conflated information to the end of the queue.

connector

A configured point of connection to a server.

There can be one or more connectors (each listening on a different port). Each connector can accept single or multiple types of connection.

consume

When a message or update is received by a topic listener and that listener chooses not to pass on the message or update to subsequent topic listeners. Topic streams cannot choose to consume messages or updates they receive.

conversation

The communication between a client and the server for a single request.

A conversation is all the communication between a client and a Diffusion server for a single request. Each conversation is identified with a `cid`.

CORS

Cross-Origin Resource Sharing

cross-origin resource sharing (CORS)

CORS

CORS

CPU

Central Processing Unit

CPU

CPU

CPU

credentials

A piece of information that is used to authenticate a principal.

CSR

Certificate Signing Request

certificate signing request (CSR)

CSR

CSR

CSS

Cascading Style Sheet

CSS

CSS

CSS

D

DAR file

A Diffusion archive file. This file contains a publisher and can be deployed on the Diffusion server.

DAR

DAR

DAR

delimiter

A byte value that is used as a separator in messages or updates.

Depending on the type of message or update, it can contain field delimiters or record delimiters.

delta

Data that is sent to a client subscribed to a topic. The delta contains only information that has been updated on the topic since the last data was sent to the client. Topics that contain only a single item of data cannot use delta messages.

Diffusion

The Diffusion product comprising the Diffusion server and client libraries.

DLL

Dynamic Link Library

DLL

DLL

DLL

DOM

Document Object Model

DOM

DOM

DOM

DMZ

De-militarized Zone

de-militarized zone (DMZ)

DMZ

DMZ

E

EULA

End User License Agreement

EULA

EULA

EULA

F

feature

An API module that contains a conceptual set of facilities.

fetch

A request from a client for the current state of all data on a topic. A client can fetch a topic's state without being subscribed to the topic. This request-response mechanism of getting data from a topic is separate from the pub-sub mechanism.

flow control

A mechanism within a Diffusion client that limits the rate that client sends messages as the load level from that client increases. A client application rapidly making thousands of calls to the Diffusion server might overflow the internal queues, which results in the client session being closed. Flow control protects against these queues overflowing by progressively delaying messages from the client to the Diffusion server.

field

A section of content that contains data of a specific type. Fields are nested inside records. A record can contain one or many fields.

G

GBE

Gigabit Ethernet

GBE

GBE

GBE

global-scoped permission

Permissions at global scope apply to actions on the Diffusion server.

GUI

Graphical User Interface

GUI

GUI

GUI

H

handler

A handler is an object responsible for responding to one or more instances of a single type of request.

HDD

Hard Disk Drive

HDD

HDD

HDD

HTML

Hypertext Markup Language

HTML

HTML

HTML

HTTP

Hypertext Transfer Protocol

HTTP

HTTP

HTTP

I

IDE

Integrated Development Environment

integrated development environment (IDE)

IDE

IDE

ISAPI

Internet Server Application Programming Interface

ISAPI

initial topic load

The data sent to a client when it first subscribes to a topic. This data contains the value of the current state of the topic.

initial topic load (ITL)

ITL

ITL

J

JAR

Java Archive

JAR

JAR

JAR

JDK

Java Development Kit

Java Development Kit (JDK)

JDK

JDK

JMS

Java Message Service

Java Message Service (JMS)

JMS

JMS

JMX

Java Management Extensions

Java Management Extensions (JMX)

JMX

JMX

JRE

Java Runtime Environment

Java Runtime Environment (JRE)

JRE

JRE

JSON

JavaScript Object Notation

JavaScript Object Notation (JSON)

JSON

JSON

JVM

Java Virtual Machine

Java Virtual Machine (JVM)

JVM

JVM

L

LDAP

Lightweight Directory Access Protocol

Lightweight Directory Access Protocol (LDAP)

LDAP

LDAP

listener

In the API, a listener is an object that is always called when a particular event occurs.

M

message

A message is a series of bytes of information formatted according to the Diffusion protocol which can be sent between components utilizing Diffusion.

message queue

A queue of messages. Each client connection to Diffusion has such a queue on the Diffusion server upon which messages are put for sending to the client.

queue

metadata

Data about data. In Diffusion metadata is used to define the structure of messages.

message metadata

multicast

To send data to several recipients at the same time.

The datagrid uses multicasting to locate other datagrid nodes.

N

NAT

Network Address Translation

network address translation (NAT)

NAT

NAT

NIC

Network Interface Controller

NIC

NIC

NIC

NIO

New Input-Output

NIO

NIO

Non-blocking Input/Output

NIO

notification

...

P

path

A string representation of a location at which a topic can exist or through which messages can be sent.

A path consists of parts separated by a slash character (/).

Paths describe a location where a topic can be bound and used for pub-sub distribution of data. When used this way they can be referred to as topic paths.

Paths can be used for bi-directional messaging. When used this way they can be referred to as message paths. A client can send a message to a message path and the Diffusion server routes the message to the message handler for the message path.

topic name

topic path

message path

hierarchic topic name

full topic name

PNG

Portable Network Graphics

PNG

PNG

PNG

permission

A permission represents the right to perform an action on the Diffusion server or on data hosted by the Diffusion server. Permissions can be global- or topic-scoped.

PID

Process ID

PID

PID

PID

ping

A query sent by a client or by the Diffusion server to a connected component to check that the connection exists and the latency of the connection.
The following types of ping are available:

server ping

A client pings the Diffusion server.

system ping

Diffusion pings all clients at a regular interval.

PDF

Portable Document Format

PDF

PDF

PDF

primary server

In a fan-out solution, the server from which updates are fanned out to replica servers.

In previous releases, this server was called the master server. This terminology in no longer used.

master server

principal

An identity that can be authenticated by the Diffusion server or by a client.

A principal functions like a username, except that instead of identifying a particular person, it denotes an identity that can be used by a person or client. For example, you can use the 'admin' principal to log in to the Diffusion console. After a principal has been authenticated, it can be assigned roles that enable it to access actions or resources.

protocol

A protocol defines the exact format of data passed between the Diffusion server and a client.

publisher

An obsolete server-side component used to publish messages to one or more topics.

Publishers were supported in previous versions of Diffusion. They are no longer supported. Publishing is now done with clients.

publishing topic

A topic where data is published and from which the data is distributed to subscribing clients.

Q

message queue

A queue of messages. Each client connection to Diffusion has such a queue on the Diffusion server upon which messages are put for sending to the client.

queue

R

RAID

Redundant Array of Independent Disks

RAID

RAID

RAID

RAM

Random Access Memory

RAM

RAM

RAM

record

A section of content that acts as a container for a set of fields. Inside the content of a message or update you can have one or many records. A record can contain one or many fields.

regular expression

A string that uses special characters to describe a search pattern.

Diffusion uses Java-style regular expressions.

regex

replica server

In a fan-out solution, a server to which updates are fanned out from the primary server.

In previous releases, this server was called the slave server. This terminology in no longer used.

slave server

RMI

Remote Method Invocation

remote method invocation (RMI)

RMI

RMI

role

A role is a named set of permissions and other roles. Principals and sessions can both be assigned roles.

role hierarchy

Roles are hierarchical. A role can include other roles and, by doing so, have the permissions assigned to the included roles. A role cannot include itself, either directly or indirectly – through a number of included roles.

RPM

Redhat Package Manager

Redhat Package Manager (RPM)

RPM

RPM

S

SAS

Serial Attached SCSI

SAS

SAS

SAS

SDK

Software Development Kit

software development kit (SDK)

SDK

SDK

server

The component that hosts topics. A server broadcasts topic updates to all subscribed clients.

Clients can connect to servers through the API.

Diffusion server

session

An ongoing dialog between a client and the Diffusion server.

Typically, a session represents a single client connection to a single server. However, in the event of connection failure the session can automatically reconnect to the same server or even fail over to another server and still retain its context.

SLF4J

Simple Logging Facade for Java

SLF4J

SLF4J

SLF4J

SSH

Secure Shell

SSH

SSH

SSH

SSL

Secure Sockets Layer

Secure Sockets Layer (SSL)

SSL

SSL

state

The latest published values of all data items on the topic. The state of a topic is stored on the Diffusion server.

stateful topic

A topic that stores a current value as topic data on the Diffusion server.

stateless topic

A deprecated topic type that does not store a current value on the Diffusion server. Replaced by the DONT_RETAIN_VALUE property.

structural conflation

A form of conflation that enables you to define the operations performed on outdated content. You can merge, aggregate, reverse or combine the effects of multiple changes into a single consistent and current notification to the client.

stream

In the API, a stream is a sequence of responses to a single request.

subscribe

A client registers interest in a topic such that the client receives messages sent to that topic.

support

Push Technology supports a number of hardware and software versions, these versions have not necessarily been tested. Those hardware and software versions that we have tested are listed as 'certified'.

T

TCP

Transmission Control Protocol

TCP

TCP

TCP

throttling

Limiting the volume of messages that the Diffusion server transmits to a client within a specified period of time.

Throttling can be used to limit bandwidth usage or to prevent more messages being sent to a client than the client can handle.

TLS

Transport Layer Security

Transport Layer Security (TLS)

TLS

TLS

topic

A logical channel through which messages are distributed.

Topics provide a logical link between publishers and subscribers. Clients publish messages to topics. Clients subscribe to topics to receive messages published to that topic.

topic path prefix

The root part of a topic selector.

A concrete topic path to the most specific part of the topic tree that contains all topics that the selector can specify. For example, for the topic selector `?foo/bar/baz/.*/bing`, the topic path prefix is `foo/bar/baz`.

path prefix

topic selector

An object that retrieves one or more topics based on their topic paths.

A topic selector uses a pattern expression, which can include one or more regular expressions, to match to the path of one of more topics.

selector

topic-scoped permission

Permissions at topic scope apply to actions on a topic.

Topic-scoped permissions are defined against topic branches. The permissions that apply to a topic are the set of permissions defined at the most specific branch of the topic tree.

topic tree

The organization structure of topics on the Diffusion server.

A topic can have subtopics and can itself be a subtopic of another topic. All topics created on the Diffusion server by a client are in the topic tree.

topic hierarchy

transport

An implementation of a network protocol. The mechanism by which clients communicate with the Diffusion server.

U

update

Data published to a topic by a client that is applied to the topic to change the topic state. The updated data is then pushed out to all subscribing clients.

URL

Uniform Resource Locator

URL

URL

URL

UTF-8

Universal Character Set Transformation Format 8-bit

A character encoding capable of encoding all possible characters in Unicode.

UTF-8

UTF-8

UTF-8

V

VCPU

Virtual Central Processing Unit

VCPU

VCPU

VCPU

W

WAR

Web Application Archive

WAR

WAR

WAR

X

XHR

XmlHttpRequest

XHR

XHR

XHR

XML

Extensible Markup Language

XML

XML

XML

XSD

XML Schema Definition

XSD

XSD

XSD

Appendix C

Trademarks

The following trademarked terms are included in this manual.

Diffusion is trademark of Push Technology Ltd.

Adobe® is a registered trademark of Adobe Systems Incorporated.

AIX™, IBM Cloud, Cast Iron®, and WebSphere® are trademarks of IBM.

Amazon™ and Amazon EC2™ are trademarks of Amazon.

Android and Chrome are trademarks of Google Inc.

Ant, Apache, Apache Derby™, Apache Tomcat™, and Maven are trademarks of The Apache Software Foundation.

Apple, Mac®, macOS, Safari, and Siri® are registered trademarks of Apple Inc.

BlackBerry® is a registered trademark of RIM.

CentOS and Red Hat are trademarks or registered trademarks of Red Hat, Inc.

Dell™ is trademark of Dell, Inc.

Docker is trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries.

Eclipse™ is a trademark of the Eclipse Foundation, Inc.

F5 is a registered trademark of F5 Networks, Inc.

Firefox is a registered trademark of Mozilla Foundation.

Hazelcast is a trademark of Hazelcast Inc.

Intel and Xeon are trademarks of Intel Corporation.

Internet Explorer, Microsoft, and Windows are trademarks or registered trademarks of Microsoft Corporation.

iOS is a registered trademark of Cisco.

Java, JavaScript, Oracle, and Solaris™ are trademarks or registered trademarks of Oracle Corporation.

Linux is a trademark of Linus Torvalds.

Nagios® is a registered trademark of Nagios Enterprises.

Node.js is a trademark of Joyent, Inc.

Opera is a registered trademark of Opera Software ASA.

Splunk is a trademark of Splunk, Inc.

SUSE® is a registered trademark of SUSE LLC.

TIBCO Enterprise Message Service™ is a trademark of TIBCO Software Inc.

Ubuntu® is a registered trademark of Canonical Ltd.

UNIX is a registered trademark of The Open Group.

VeriSign® is a registered trademark of VeriSign, Inc.

VMware® and VMware vSphere are registered trademarks of VMware, Inc.

Appendix D

Copyright Notices

Diffusion uses third party, open source software. The rights to this software are not owned by Push Technology and the software is distributed under different licensing agreements. The distribution and use of third-party software is subject to the applicable terms.

The following sections list the software used, their licenses, copyright notices and disclaimers.

In this section:

- ANTLR
- Bouncy Castle
- Apache Commons Codec
- Apache Portable Runtime
- Bootstrap
- CQEngine
- cron4j
- d3
- disruptor
- Fluidbox
- GeoIP2 API
- GeoLite2 City Database
- GeoIP2 API
- geronimo-jms_1.1_spec
- Google code prettify
- hashmap
- Hazelcast
- HPPC
- htmlcompressor
- inherits
- iStack Common Runtime
- jackson-annotations
- jackson-core

- jackson-dataformat-cbor
- jackson-databind
- JAXB
- JCIP Annotations
- JCTools
- jQuery
- jquery.floatThead
- json-simple
- Knockout
- libwebsockets
- License3j
- log4j2
- loglevel
- long
- Metrics
- Minimal JSON
- Modernizr
- NLog
- opencsv
- OpenSSL
- PCRE
- Picocontainer
- Prometheus Java Simpleclient
- Rickshaw
- Servlet API
- SLF4J
- slf4j-android-logger
- SocketRocket
- streamsupport
- Tabber
- Tapestry (Plastic)
- when
- ws
- Licenses

ANTLR

Version 4.7.1

http://www.antlr.org

ANTLR is distributed under the BSD 3-clause License.

Copyright (c) 2014 Terence Parr, Sam Harwell

Bouncy Castle

Version 1.52

http://www.bouncycastle.org/java.html

Bouncy Castle is distributed under the MIT License.

Copyright (c) 2000-2015 The Legion Of The Bouncy Castle

Apache Commons Codec

Version 1.9

http://commons.apache.org/codec/

Apache Commons Codec is distributed under the Apache License 2.0.

Copyright 2002-2011 The Apache Software Foundation. All Rights Reserved

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
Apache Commons Codec
Copyright 2002-2011 The Apache Software Foundation

This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

---------------------------------------------------------------------------
src/test/org/apache/commons/codec/language/DoubleMetaphoneTest.java
 contains
test data from http://aspell.sourceforge.net/test/batch0.tab.

Copyright (C) 2002 Kevin Atkinson (kevina@gnu.org). Verbatim copying
and distribution of this entire article is permitted in any medium,
provided this notice is preserved.
---------------------------------------------------------------------------
```

Apache Portable Runtime

Version 1.5.3 (patched by Push Technology)

http://apr.apache.org

APR is distributed under the Apache 2.0 License.

Copyright (c) 2015 The Apache Software Foundation

Bootstrap

Version: 3.2.0

https://github.com/twbs/bootstrap/

Bootstrap is distributed under the MIT License.

Copyright (c) 2011-2014 Twitter, Inc

Additional notes

We also use Glyphicons, which are included as part of Bootstrap.

CQEngine

Version 3.0.0

https://github.com/npgall/cqengine

CQEngine is distributed under the Apache 2.0 License.

Copyright 2012-2015 Niall Gallagher

cron4j

Version 2.2.5

http://www.sauronsoftware.it/projects/cron4j/

Cron4j is distributed under the LGPL 2.1.

Copyright (C) 2007-2010 Carlo Pelliccia (www.sauronsoftware.it)

Source code is available from the following location: http://sourceforge.net/projects/cron4j/files/cron4j/2.2.5/cron4j-2.2.5.zip/download/

For a fee, Push Technology can also provide this source on a CD. To request a copy, contact support@pushtechnology.com.

d3

Version 3.5.16

http://d3js.org/

d3 is distributed under the BSD 3-clause License.

Copyright (c) 2010-2014, Michael Bostock

disruptor

Version 3.4.2

https://github.com/LMAX-Exchange/disruptor

disruptor is distributed under the Apache License 2.0.

Copyright 2011 LMAX Ltd.

Fluidbox

https://github.com/terrymun/Fluidbox

Fluidbox is distributed under the MIT License.

Copyright (c) 2014 Terry Mun

GeoIP2 API

Version 2.12.0

http://www.maxmind.com/en/opensource

The GeoIP2 API is distributed under the Apache 2.0 License.

Copyright (c) 2013-2018 MaxMind Inc.

GeoLite2 City Database

http://dev.maxmind.com/geoip/geoip2/geolite2/

The GeoLite City Database is distributed under the Creative Commons Attribution-ShareAlike 4.0 International License.

Copyright MaxMind, Inc.

GeoIP2 API

Version 2.12.0

http://www.maxmind.com/en/opensource

The GeoIP2 API is distributed under the Apache 2.0 License.

Copyright (c) 2013-2018 MaxMind Inc.

geronimo-jms_1.1_spec

Version 1.1

http://geronimo.apache.org/

geronimo-jms_1.1_spec is distributed under the Apache License 2.0

Copyright 2003-2006 The Apache Software Foundation

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
Apache Geronimo
Copyright 2003-2006 The Apache Software Foundation

This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).
```

Google code prettify

https://github.com/google/code-prettify

Prettify is distributed under the Apache 2.0 License.

Copyright (C) 2006 Google Inc.

hashmap

Version: 2.0.3

https://github.com/flesler/hashmap

hashmap is distributed under the MIT License.

Copyright (c) 2012-2013 Ariel Flesler aflesler@gmail.com

Hazelcast

Version 3.12.4

http://www.hazelcast.org/

Hazelcast is distributed under the Apache License 2.0

Copyright (c) 2008-2019, Hazelcast, Inc. All Rights Reserved.

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
The packages:

com.hazelcast.util.collection
com.hazelcast.internal.util.concurrent

and the classes:

com.hazelcast.util.QuickMath
com.hazelcast.client.impl.protocol.util.UnsafeBuffer
com.hazelcast.client.impl.protocol.util.BufferBuilder

contain code originating from the Agrona project
(https://github.com/real-logic/Agrona).

The class com.hazelcast.util.HashUtil contains code originating
from the Koloboke project (https://github.com/OpenHFT/Koloboke).
```

```
The class classloading.ThreadLocalLeakTestUtils contains code
 originating
from the Tomcat project (https://github.com/apache/tomcat).

com.hazelcast.internal.cluster.fd.PhiAccrualFailureDetector contains
 code originating
from the Akka project (https://github.com/akka/akka/).

The package com.hazelcast.internal.json contains code originating
from minimal-json project (https://github.com/ralfstx/minimal-json).
```

HPPC

Version 0.8.1

http://labs.carrotsearch.com/hppc.html

HPPC is distributed under the Apache License 2.0.

Copyright 2010-2013, Carrot Search s.c., Boznicza 11/56, Poznan, Poland

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
ACKNOWLEDGEMENT
===============

HPPC borrowed code, ideas or both from:

 * Apache Lucene, http://lucene.apache.org/
   (Apache license)
 * Fastutil, http://fastutil.di.unimi.it/
   (Apache license)
 * Koloboke, https://github.com/OpenHFT/Koloboke
   (Apache license)
```

htmlcompressor

Version 1.5.2

http://code.google.com/p/htmlcompressor/

The htmlcompressor is distributed under the Apache License 2.0.

Copyright 2009-2011 Sergiy Kovalchuk

Additional notices: Apache License 2.0 Notice

inherits

Version 2.0.1

https://github.com/isaacs/inherits

Inherits is distributed under the ISC License.

Copyright (c) Isaac Z. Schlueter.

iStack Common Runtime

Version 3.0.7

https://github.com/javaee/jaxb-istack-commons

JAXB is distributed under the CDDL 1.1.

Copyright (c) 1997-2012 Oracle and/or its affiliates

jackson-annotations

Version 2.9.7

https://github.com/FasterXML/jackson-annotations

jackson-annotations is distributed under the Apache License 2.0

Copyright 2009-2011 FasterXML

jackson-core

Version 2.9.7

https://github.com/fasterxml/jackson-core

jackson-core is distributed under the Apache License 2.0

Copyright Tatu Saloranta

jackson-dataformat-cbor

Version 2.9.7

https://github.com/FasterXML/jackson-dataformats-binary

jackson-dataformat-cbor is distributed under the Apache License 2.0

Copyright Tatu Saloranta

jackson-databind

Version 2.9.7

https://github.com/FasterXML/jackson-databind

jackson-databind is distributed under the Apache License 2.0

Copyright Tatu Saloranta

JAXB

Version 2.3.1

https://github.com/javaee/jaxb-v2

JAXB is distributed under the CDDL 1.1.

Copyright (c) 1997-2012 Oracle and/or its affiliates

JCIP Annotations

Version 1

https://github.com/stephenc/jcip-annotations

jcip-annotations is distributed under the Apache License 2.0

Copyright 2013 Stephen Connolly.

JCTools

Version 2.1.2

https://github.com/JCTools/JCTools

JCTools is distributed under the Apache License 2.0

Copyright 2016 Nitsan Wakart.

jQuery

Version: 1.7.1

https://jquery.org/

jQuery is distributed under the MIT License.

Copyright 2014 jQuery Foundation and other contributors http://jquery.com/

jquery.floatThead

Version: 2.0.3

https://mkoryak.github.io/floatThead/

jquery.floatThead is distributed under the MIT License.

Copyright 2012-2017 Misha Koryak

json-simple

Version 1.1.1

https://github.com/fangyidong/json-simple

json-simple is distributed under the Apache License 2.0.

Copyright (c) Yidong Fang, Chris Nokleberg

Knockout

Version 2.1.0

http://knockoutjs.com/

Knockout is distributed under the MIT License.

Copyright (c) Steven Sanderson, the Knockout.js team, and other contributors

libwebsockets

Version 1.7.7

https://libwebsockets.org/index.html

libwebsockets is distributed under the LGPL 2.1.

Copyright (C) 2010-2015 Andy Green <andy@warmcat.com>

Source code is available from the following location: https://github.com/warmcat/libwebsockets

For a fee, Push Technology can also provide this source on a CD. To request a copy, contact support@pushtechnology.com.

License3j

Version 1.0.7

https://github.com/verhas/License3j

This version of Licence3j was distributed under the Apache License 2.0.

Copyright Peter Verhas

log4j2

Version 2.11.1

http://logging.apache.org/log4j/2.x/

log4j2 is distributed under the Apache License 2.0.

Copyright The Apache Software Foundation. All Rights Reserved.

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
Apache Log4j
Copyright 1999-2017 Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

ResolverUtil.java
Copyright 2005-2006 Tim Fennell
```

```
Dumbster SMTP test server
Copyright 2004 Jason Paul Kitchen

TypeUtil.java
Copyright 2002-2012 Ramnivas Laddad, Juergen Hoeller, Chris Beams
```

loglevel

Version: 1.4.0

https://github.com/pimterry/loglevel

loglevel is distributed under the MIT License.

Copyright (c) 2013 Tim Perry

long

Version: 2.2.5

https://github.com/dcodeIO/Long.js

long is distributed under the Apache License 2.0.

Copyright 2013 Daniel Wirtz dcode@dcode.io

Copyright 2009 The Closure Library Authors. All Rights Reserved.

Metrics

Version 3.0.0-BETA

http://metrics.codahale.com/

Metrics is distributed under the Apache License 2.0.

Copyright (c) 2010-2013 Coda Hale, Yammer.com

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
Metrics
Copyright 2010-2013 Coda Hale and Yammer, Inc.

This product includes software developed by Coda Hale and Yammer, Inc.

This product includes code derived from the JSR-166 project
 (ThreadLocalRandom, Striped64,
LongAdder), which was released with the following comments:

    Written by Doug Lea with assistance from members of JCP JSR-166
    Expert Group and released to the public domain, as explained at
    http://creativecommons.org/publicdomain/zero/1.0/
```

Minimal JSON

https://github.com/ralfstx/minimal-json

Minimal JSON is distributed under the MIT License.

Copyright (c) 2014, 2015 EclipseSource

Modernizr

Version: 2.8.3

http://modernizr.com/

Modernizr is distributed under the MIT License and BSD 3-clause License.

NLog

Version 4.0.0

https://github.com/NLog/NLog/

NLog is distributed under the BSD 3-clause License.

Copyright (c) 2004-2011 Jaroslaw Kowalski <jaak@jkowalski.net>

opencsv

Version 2.3

http://opencsv.sourceforge.net/

opencsv is distributed under the Apache License 2.0.

Copyright 2005 Bytecode Pty Ltd.

OpenSSL

Version 1.0.2a

https://www.openssl.org/

OpenSSL is distributed under the OpenSSL and SSLeay Licenses.

PCRE

Version 1.5.2

http://www.pcre.org/

PCRE is distributed under the BSD 3-clause License.

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel

Email local part: ph10

Email domain: cam.ac.uk

University of Cambridge Computing Service,

Cambridge, England.

Copyright (c) 1997-2015 University of Cambridge

All rights reserved.

PCRE2 JUST-IN-TIME COMPILATION SUPPORT

Written by: Zoltan Herczeg

Email local part: hzmester

Emain domain: freemail.hu

Copyright(c) 2010-2015 Zoltan Herczeg

All rights reserved.

STACK-LESS JUST-IN-TIME COMPILER

Written by: Zoltan Herczeg

Email local part: hzmester

Emain domain: freemail.hu

Copyright(c) 2009-2015 Zoltan Herczeg

All rights reserved.

Picocontainer

Version 2.15

http://picocontainer.codehaus.org/

Picocontainer is distributed under the BSD 3-clause License.

Copyright (c) 2003-2008 PicoContainer Organization. All rights reserved.

Prometheus Java Simpleclient

Version 0.0.50

https://github.com/prometheus/client_java/

Prometheus Java Simpleclient is distributed under the Apache License 2.0.

Copyright (c) 2012-2015 The Prometheus Authors

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
Prometheus instrumentation library for JVM applications
Copyright 2012-2015 The Prometheus Authors

This product includes software developed at
Boxever Ltd. (http://www.boxever.com/).

This product includes software developed at
SoundCloud Ltd. (http://soundcloud.com/).

This product includes software developed as part of the
Ocelli project by Netflix Inc. (https://github.com/Netflix/ocelli/).
```

Rickshaw

http://code.shutterstock.com/rickshaw/

Rickshaw is distributed under the MIT License.

Copyright (C) 2011-2013 by Shutterstock Images, LLC

Servlet API

http://jetty.mortbay.org/project/modules/servlet-api-2.5

Servlet API is distributed under the CDDL v1.0 License.

SLF4J

Version 1.7.30

http://www.slf4j.org/

SLF4J is distributed under the MIT License.

Copyright (c) 2004-2017 QOS.ch All rights reserved.

slf4j-android-logger

Version 1.0.5

https://github.com/PSDev/slf4j-android-logger

slf4j-android-logger is distributed under the Apache 2.0 License.

Copyright 2013-2016 Philip Schiffer

SocketRocket

Version 0.3.1-beta2

https://github.com/square/SocketRocket

SocketRocket is distributed under the Apache License 2.0.

Copyright 2012 Square Inc.

streamsupport

Version 1.6.0

https://github.com/streamsupport/streamsupport/

Streamsupport is distributed under the GPL 2 CE.

Source code is available from the following location: https://github.com/streamsupport/streamsupport/

For a fee, Push Technology can also provide this source on a CD. To request a copy, contact support@pushtechnology.com.

Tabber

Version: 1.9

http://www.barelyfitz.com/projects/tabber/

Tabber is distributed under the MIT License.

Copyright (c) 2006 Patrick Fitzgerald pat@barelyfitz.com

Tapestry (Plastic)

Version 5.4.3

http://tapestry.apache.org/

Tapestry is distributed under the Apache License 2.0.

Copyright 2011, 2012 The Apache Software Foundation

Additional notices

The following information is included in the `NOTICE.txt` file that accompanies the source:

```
This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

Please refer to the NOTICE.txt in each sub-module to
identify further dependencies.

The Maven central repository is the preferred method to download
 Tapestry
and its dependencies. The binary archive includes just basic
dependencies for tapestry-core; using other modules (such as
tapestry-hibernate or any of the others) requires downloading
additional dependencies. Please refer to the Maven POM for each module
to identify its dependencies.
```

The following information is included in the `plastic/NOTICE.txt` file that accompanies the source:

```
This product includes software developed by
The Apache Software Foundation (http://www.apache.org/).

This product imports and repackages ASM 5.0.1 code which is
released under a BSD style License
```

```
http://asm.ow2.org/license.html
```

when

Version 3.7.3

https://github.com/cujojs/when

http://cujojs.com/

When is distributed under the MIT License.

Copyright (c) 2011 Brian Cavalier

ws

Version 0.8.0

https://github.com/websockets/ws

WS is distributed under the MIT License.

Copyright (c) 2011 Einar Otto Stangvik

Licenses

The following licenses are used by the third party, open source software that is distributed with Diffusion.

Apache License 2.0

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display

generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Related concepts

Apache Commons Codec on page 702

Apache Portable Runtime on page 702

CQEngine on page 703

d3 on page 703

disruptor on page 703

geronimo-jms_1.1_spec on page 704

Hazelcast on page 705

HPPC on page 706

htmlcompressor on page 706

jackson-core on page 707

jackson-dataformat-cbor on page 707

JCIP Annotations on page 708

JCTools on page 708

json-simple on page 708

log4j2 on page 709

long on page 710

Metrics on page 710

Google code prettify on page 705

opencsv on page 711

slf4j-android-logger on page 713

SocketRocket on page 713

Tapestry (Plastic) on page 714

BSD 3-clause License

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

> **Note:** The copyright statement above is included in its completed form in the sections of this document specific to the individual products covered by this license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT, INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Related concepts

ANTLR on page 702

NLog on page 711

Modernizr on page 711

PCRE on page 711

Picocontainer on page 712

Common Development and Distribution License

Sun Microsystems, Inc. ("Sun") ENTITLEMENT for SOFTWARE

Licensee/Company: Entity receiving Software.

Effective Date: Date of delivery of the Software to You.

Software: JavaMail 1.4.

License Term: Perpetual (subject to termination under the SLA).

Licensed Unit: Software Copy.

Licensed unit Count: Unlimited.

Permitted Uses:

1. You may reproduce and use the Software for Individual, Commercial, or Research and Instructional Use for the purposes of designing, developing, testing, and running Your applets and application("Programs").

2. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software's documentation, You may reproduce and distribute portions of Software identified as a redistributable in the documentation ("Redistributable"), provided that:

(a) you distribute Redistributable complete and unmodified and only bundled as part of Your Programs,

(b) your Programs add significant and primary functionality to the Redistributable,

(c) you distribute Redistributable for the sole purpose of running your Programs,

(d) you do not distribute additional software intended to replace any component(s) of the Redistributable,

(e) you do not remove or alter any proprietary legends or notices contained in or on the Redistributable.

(f) you only distribute the Redistributable subject to a license agreement that protects Sun's interests consistent with the terms contained in this Agreement, and

(g) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Redistributable.

3. Java Technology Restrictions. You may not create, modify, or change the behavior of, or authorize your licensees to create, modify, or change the behavior of, classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

B. Sun Microsystems, Inc. ("Sun") SOFTWARE LICENSE AGREEMENT

READ THE TERMS OF THIS AGREEMENT ("AGREEMENT") CAREFULLY BEFORE OPENING SOFTWARE MEDIA PACKAGE. BY OPENING SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" (OR "EXIT") BUTTON AT THE END OF THIS AGREEMENT. IF YOU HAVE SEPARATELY AGREED

TO LICENSE TERMS ("MASTER TERMS") FOR YOUR LICENSE TO THIS SOFTWARE, THEN SECTIONS 1-5 OF THIS AGREEMENT ("SUPPLEMENTAL LICENSE TERMS") SHALL SUPPLEMENT AND SUPERSEDE THE MASTER TERMS IN RELATION TO THIS SOFTWARE.

1. Definitions.

(a) "Entitlement" means the collective set of applicable documents authorized by Sun evidencing your obligation to pay associated fees (if any) for the license, associated Services, and the authorized scope of use of Software under this Agreement.

(b) "Licensed Unit" means the unit of measure by which your use of Software and/or Service is licensed, as described in your Entitlement.

(c) "Permitted Use" means the licensed Software use(s) authorized in this Agreement as specified in your Entitlement. The Permitted Use for any bundled Sun software not specified in your Entitlement will be evaluation use as provided in Section 3.

(d) "Service" means the service(s) that Sun or its delegate will provide, if any, as selected in your Entitlement and as further described in the applicable service listings at www.sun.com/service/servicelist.

(e) "Software" means the Sun software described in your Entitlement. Also, certain software may be included for evaluation use under Section 3.

(f) "You" and "Your" means the individual or legal entity specified in the Entitlement, or for evaluation purposes, the entity performing the evaluation.

2. License Grant and Entitlement.

Subject to the terms of your Entitlement, Sun grants you a nonexclusive, nontransferable limited license to use Software for its Permitted Use for the license term. Your Entitlement will specify (a) Software licensed, (b) the Permitted Use, (c) the license term, and (d) the Licensed Units.

Additionally, if your Entitlement includes Services, then it will also specify the (e) Service and (f) service term.

If your rights to Software or Services are limited in duration and the date such rights begin is other than the purchase date, your Entitlement will provide that beginning date(s).

The Entitlement may be delivered to you in various ways depending on the manner in which you obtain Software and Services, for example, the Entitlement may be provided in your receipt, invoice or your contract with Sun or authorized Sun reseller. It may also be in electronic format if you download Software.

3. Permitted Use.

As selected in your Entitlement, one or more of the following Permitted Uses will apply to your use of Software. Unless you have an Entitlement that expressly permits it, you may not use Software for any of the other Permitted Uses. If you don't have an Entitlement, or if your Entitlement doesn't cover additional software delivered to you, then such software is for your Evaluation Use.

(a) Evaluation Use. You may evaluate Software internally for a period of 90 days from your first use.

(b) Research and Instructional Use. You may use Software internally to design, develop and test, and also to provide instruction on such uses.

(c) Individual Use. You may use Software internally for personal, individual use.

(d) Commercial Use. You may use Software internally for your own commercial purposes.

(e) Service Provider Use. You may make Software functionality accessible (but not by providing Software itself or through outsourcing services) to your end users in an extranet deployment, but not to your affiliated companies or to government agencies.

4. Licensed Units.

Your Permitted Use is limited to the number of Licensed Units stated in your Entitlement. If you require additional Licensed Units, you will need additional Entitlement(s).

5. Restrictions.

(a) The copies of Software provided to you under this Agreement are licensed, not sold, to you by Sun. Sun reserves all rights not expressly granted. (b) You may make a single archival copy of Software, but otherwise may not copy, modify, or distribute Software. However if the Sun documentation accompanying Software lists specific portions of Software, such as header files, class libraries, reference source code, and/or redistributable files, that may be handled differently, you may do so only as provided in the Sun documentation. (c) You may not rent, lease, lend or encumber Software. (d) Unless enforcement is prohibited by applicable law, you may not decompile, or reverse engineer Software. (e) The terms and conditions of this Agreement will apply to any Software updates, provided to you at Sun's discretion, that replace and/or supplement the original Software, unless such update contains a separate license. (f) You may not publish or provide the results of any benchmark or comparison tests run on Software to any third party without the prior written consent of Sun. (g) Software is confidential and copyrighted. (h) Unless otherwise specified, if Software is delivered with embedded or bundled software that enables functionality of Software, you may not use such software on a stand-alone basis or use any portion of such software to interoperate with any program(s) other than Software. (i) Software may contain programs that perform automated collection of system data and/or automated software updating services. System data collected through such programs may be used by Sun, its subcontractors, and its service delivery partners for the purpose of providing you with remote system services and/or improving Sun's software and systems. (j) Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility and Sun and its licensors disclaim any express or implied warranty of fitness for such uses. (k) No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement.

6. Term and Termination.

The license and service term are set forth in your Entitlement(s). Your rights under this Agreement will terminate immediately without notice from Sun if you materially breach it or take any action in derogation of Sun's and/or its licensors' rights to Software. Sun may terminate this Agreement should any Software become, or in Sun's reasonable opinion likely to become, the subject of a claim of intellectual property infringement or trade secret misappropriation. Upon termination, you will cease use of, and destroy, Software and confirm compliance in writing to Sun. Sections 1, 5, 6, 7, and 9-15 will survive termination of the Agreement.

7. Java Compatibility and Open Source.

Software may contain Java technology. You may not create additional classes to, or modifications of, the Java technology, except under compatibility requirements available under a separate agreement available at www.java.net.

Sun supports and benefits from the global community of open source developers, and thanks the community for its important contributions and open standards-based technology, which Sun has adopted into many of its products.

Please note that portions of Software may be provided with notices and open source licenses from such communities and third parties that govern the use of those portions, and any licenses granted hereunder do not alter any rights and obligations you may have under such open source licenses, however, the disclaimer of warranty and limitation of liability provisions in this Agreement will apply to all Software in this distribution.

8. Limited Warranty.

Sun warrants to you that for a period of 90 days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to

replace Software media or refund the fee paid for Software. Some states do not allow limitations on certain implied warranties, so the above may not apply to you. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

9. Disclaimer of Warranty.

UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

10. Limitation of Liability.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose. Some states do not allow the exclusion of incidental or consequential damages, so some of the terms above may not be applicable to you.

11. Export Regulations.

All Software, documents, technical data, and any other materials delivered under this Agreement are subject to U.S. export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with these laws and regulations and acknowledge that you have the responsibility to obtain any licenses to export, re-export, or import as may be required after delivery to you.

12. U.S. Government Restricted Rights.

If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

13. Governing Law.

Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

14. Severability.

If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

15. Integration.

This Agreement, including any terms contained in your Entitlement, is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

Please contact Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95054 if you have questions.

Related concepts

Servlet API on page 713

Eclipse Public License – v 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

- a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
- b) in the case of each subsequent Contributor:
  - i) changes to the Program, and
  - ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program. "Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

- a) it complies with the terms and conditions of this Agreement; and
- b) its license agreement:
  - i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
  - ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
  - iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
  - iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

- a) it must be made available under this Agreement; and
- b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

GNU General Public License, version 2, with the Classpath Exception

The GNU General Public License (GPL)

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software – to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items – whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL Certain source files distributed by Oracle America and/or its affiliates are subject to the following clarification and special exception to the GPL, but only where Oracle has expressly included in the particular source file's header the words "Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the LICENSE file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

Related concepts

streamsupport on page 714

ISC License –

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Related concepts

inherits on page 706

The GNU Lesser General Public License, version 2.1 (LGPL-2.1)

GNU Lesser General Public License

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software – to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages – typically libraries – of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Related concepts

cron4j on page 703

libwebsockets on page 709

The MIT License (MIT)

Copyright (c) <year> <copyright holders>

Note: The copyright statement above is included in its completed form in the sections of this document specific to the individual products covered by this license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Related concepts

Bootstrap on page 703

Fluidbox on page 704

hashmap on page 705

jQuery on page 708

Knockout on page 709

loglevel on page 710

Modernizr on page 711

Rickshaw on page 713

SLF4J on page 713

Tabber on page 714

Minimal JSON on page 711

when on page 715

ws on page 715

OpenSSL and SSLeay Licenses

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts.

OpenSSL License

Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)

All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)"

   The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Related concepts

OpenSSL on page 711