MigratoryData Client API for ReactPHP

Developer's Guide and Reference Manual

July 6, 2020

Contents

Chapter 1

Developer's Guide

This guide includes the following sections:

1.1 Overview

This Application Programming Interface (API) contains all the necessary operations for connecting to a cluster of one or more MigratoryData servers, subscribing to one or more subjects, getting real-time messages for the subscribed subjects, and publishing real-time messages.

Before reading this manual, it is recommended to read *MigratoryData Architecture Guide* (PDF, HTML).

1.2 Creating PHP clients for MigratoryData Server

A typical API usage is as follows:

1.2.1 Step 1 - Include the libraries

Use the following to include the library:

```
require_once '../../lib/migratorydata-client.php';
```

Use composer to install react event-loop and react socket. After that import the autoload.php file.

```
require 'vendor/autoload.php';
```

1.2.2 Step 2 - Create react event loop and give the loop to the client

Use the API call `MigratoryDataClient.setLoop()` to attach the created react event loop.

1.2.3 Step 3 - Define a listener to get messages and status notifications

The listener should implement the `MigratoryDataListener` interface.

Use the API call `MigratoryDataClient.setListener()` to attach your listener implementation.

1.2.4 Step 4 - Specify where to connect to

Specify a cluster of one or more MigratoryData servers to which the Java client will connect to using the API method `MigratoryDataClient.setServers()`. In fact, the client will connect to only one of the MigratoryData servers in this list. But, defining two or more MigratoryData servers is recommended in order to achieve fail-over. Supposing the MigratoryData server to which the client connected goes down, then the API will automatically reconnect the client to another MigratoryData server in the list.

1.2.5 Step 5 - Connect to the MigratoryData cluster

Use the API method `MigratoryDataClient.connect()` to connect to the cluster and start receiving real-time messages from the MigratoryData cluster as well as status notifications.

1.2.6 Step 6 - Subscribe to subjects and publish messages

Use the API method `MigratoryDataClient.subscribe()` to specify interest in receiving real-time messages having as subjects the strings provided in the parameter of this API method. You can call the API method `MigratoryDataClient.subscribe()` at any time to subscribe to further subjects. To unsubscribe from subscribed subjects, use the API method `MigratoryDataClient.unsubscribe()`.

Use the API method `MigratoryDataClient.publish()` to publish messages.

1.2.7 Step 7 - Handle the real-time messages and status notifications

Handle the messages received for the subscribed subjects as well as the status notifications in your listener implementation defined at Step 2 above.

Use the API method `MigratoryDataClient.publish()` to publish messages.

1.3 Examples

Examples built with this API are available in the folder `examples` of this API package; start with the README file which explains how to run them.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 MigratoryDataClient Class Reference

This class implements all the necessary operations for connecting to a cluster of one or more MigratoryData servers, subscribing to one or more subjects, getting real-time messages for the subscribed subjects, and publishing messages.

Public Member Functions

- MigratoryDataClient ()

  Create a MigratoryDataClient object.
- void setLoop ()

  Attach the application react event loop.
- void connect ()

  Use this method to connect the client to one of the MigratoryData servers specified with MigratoryDataClient.setServers(), and subscribe to the subjects specified with MigratoryDataClient.subscribe(), if any.
- void setLogListener (MigratoryDataLogListener logListener, MigratoryDataLogLevel logLevel)

  Attach a listener for handling log messages outputted by the library.
- void setListener (MigratoryDataListener listener)

  Attach a listener for handling the received real-time messages as well as the status notifications.
- MigratoryDataListener getListener ()

  Get the listener for handling real-time messages and status notifications.
- void setServers (array< String > servers)

  Specify a cluster of one or more MigratoryData servers to which the client will connect to.
- void subscribe (array< String > subjects)

  Subscribe to one or more subjects.
- void subscribeWithConflation (array< String > subjects, int conflationMillis)

  Subscribe to one or more subjects with conflation.
- void subscribeWithHistory (array< String > subjects, int numberOfHistoricalMessages)

  Subscribe to one or more subjects after getting historical messages for those subjects.
- void unsubscribe (array< String > subjects)

  Unsubscribe from one or more subjects.
- void setEncryption (boolean encrypted)

  Configure whether to use SSL/TLS encryption when connecting to a MigratoryData server.
- void setEntitlementToken (String token)

Assign an entitlement token to the client.

- Collection< String > getSubjects ()

  Return the list of subscribed subjects.

- void notifyAfterFailedConnectionAttempts (int retries)

  Define the number of failed attempts to connect to one or more MigratoryData servers before triggering a status notification MigratoryDataClient.NOTIFY_SERVER_DOWN.

- void disconnect ()

  Disconnect from the connected MigratoryData server and dispose the resources used by the connection.

- void publish (MigratoryDataMessage message)

  Publish a message.

- void setQuickReconnectMaxRetries (int retries)

  Define the maximum number of retries for the Quick Reconnect fail-over phase.

- void setQuickReconnectInitialDelay (int seconds)

  Define the number of seconds to wait before attempting to reconnect to the cluster after a connection failure is detected.

- void setReconnectPolicy (String policy)

  Define the reconnect policy to be used after the Quick Reconnect phase.

- void setReconnectTimeInterval (int seconds)

  Define the time interval used for the reconnect schedule after the Quick Reconnect phase.

- void setReconnectMaxDelay (int seconds)

  Define the maximum reconnect delay for the MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF policy.

- void setTransport (String type)

  Define the transport type used by the client to communicate with the MigratoryData cluster.

- void setConnectionTimeout (long connectionTimeoutMs)

  Define the timeout value to establish a connection with a MigratoryData server.

Static Public Attributes

- static final String NOTIFY_SERVER_UP = "NOTIFY_SERVER_UP"

  A constant which indicates that the client successfully connected to a MigratoryData server.

- static final String NOTIFY_SERVER_DOWN = "NOTIFY_SERVER_DOWN"

  A constant which indicates that the client failed to connect to a MigratoryData server.

- static final String NOTIFY_DATA_SYNC = "NOTIFY_DATA_SYNC"

  A constant which indicates that after a failover reconnection, the client successfully synchronized a subscribed subject with the latest retained message available for that subject, as well as with all messages made available during the failover period for that subject.

- static final String NOTIFY_DATA_RESYNC = "NOTIFY_DATA_RESYNC"

  A constant which indicates that after a failover reconnection, the client successfully synchronized a subscribed subject with the latest retained message available for that subject, but not with the potential messages made available during the failover period, therefore behaving as a new client.

- static final String NOTIFY_SUBSCRIBE_ALLOW = "NOTIFY_SUBSCRIBE_ALLOW"

  A constant which indicates that the client was authorized to subscribe to a subject.

- static final String NOTIFY_SUBSCRIBE_DENY = "NOTIFY_SUBSCRIBE_DENY"

  A constant which indicates that the client was not authorized to subscribe to a subject.

- static final String NOTIFY_PUBLISH_OK = "NOTIFY_PUBLISH_OK"

  A constant which indicates that the client successfully published a message.

- static final String NOTIFY_PUBLISH_FAILED = "NOTIFY_PUBLISH_FAILED"

  A constant which indicates that the client was unable to publish a message.

- static final String NOTIFY_PUBLISH_DENIED = "NOTIFY_PUBLISH_DENIED"

  A constant which indicates that the client was unable to publish a message because it is not allowed by the entitlement system.

- static final String CONSTANT_WINDOW_BACKOFF = "CONSTANT_WINDOW_BACKOFF"

  A constant used to define the reconnect policy.
- static final String TRUNCATED_EXPONENTIAL_BACKOFF = "TRUNCATED_EXPONENTIAL_BACKOFF"

  A constant used to define the reconnect policy.
- static String TRANSPORT_HTTP = "TRANSPORT_HTTP"

  A constant used to define the transport type.
- static String TRANSPORT_WEBSOCKET = "TRANSPORT_WEBSOCKET"

  A constant used to define the transport type.

3.1.1 Detailed Description

This class implements all the necessary operations for connecting to a cluster of one or more MigratoryData servers, subscribing to one or more subjects, getting real-time messages for the subscribed subjects, and publishing messages.

3.1.2 Member Function Documentation

3.1.2.1 void MigratoryDataClient.setLoop ( )

Attach the application react event loop.

Parameters

loop	an implementation of React\EventLoop\LoopInterface.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.

3.1.2.2 void MigratoryDataClient.connect ( )

Use this method to connect the client to one of the MigratoryData servers specified with MigratoryDataClient.setServers(), and subscribe to the subjects specified with MigratoryDataClient.subscribe(), if any.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_NOT_SET.
MigratoryDataException	Exceptions with the error code E_INVALID_URL.

3.1.2.3 void MigratoryDataClient.setLogListener (MigratoryDataLogListener *logListener,* MigratoryDataLogLevel *logLevel*)

Attach a listener for handling log messages outputted by the library.

It is advisable to configure the listener first to log as much as possible. If no log listener is set then, by default the client will log to the console.

Parameters

logListener	an implementation of the MigratoryDataLogListener interface
logLevel	a particular MigratoryDataLogLevel configured as the logging threshold

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.
MigratoryDataException	Exceptions with the error code E_RUNNING.

3.1.2.4 void MigratoryDataClient.setListener (MigratoryDataListener *listener*)

Attach a listener for handling the received real-time messages as well as the status notifications.

Parameters

listener	an implementation of the MigratoryDataListener interface

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.

3.1.2.5 MigratoryDataListener MigratoryDataClient.getListener ( )

Get the listener for handling real-time messages and status notifications.

This is the listener set with MigratoryDataClient.setListener().

Returns

the listener for handling real-time messages and status notifications

3.1.2.6 void MigratoryDataClient.setServers (array< String > *servers*)

Specify a cluster of one or more MigratoryData servers to which the client will connect to.

For example, to connect to a cluster formed of two MigratoryData servers installed at the addresses `p1.example.com` and `p2.example.com`, and configured to accept clients on the standard HTTP port `80`, the following code can be used:

```
client.setServers(array("p1.example.com:80", "p2.example.com:80"));
```

To achieve load-balancing, the API connects the client to a MigratoryData server chosen randomly from the `servers` list. In this way, the load is balanced among all the members of the cluster.

Moreover, the API supports weighted load-balancing. This feature is especially useful if the MigratoryData servers of the cluster are installed on machines with different capacities. You can assign to each member of the cluster a *weight* ranging from `0` to `100`. This weight assignment is a hint provided to the API to select with a higher probability a MigratoryData server with a higher weight either initially when the client connects to the cluster or later during a failover reconnection.

Supposing the address `p1.example.com` corresponds to a machine that is twice more powerful than the machine having the address `p2.example.com`, then you can assign to `p1.example.com` a weight `100` and to `p2.example.com` a weight `50` by prefixing each address with the assigned weight as follows:

```
client.setServers(array("100 p1.example.com:80", "50 p2.example.com:80"));
```

The API assigns a default weight `100` to the addresses not prefixed with a specific weight.

To achieve failover, if the connection between the client and a MigratoryData server is broken, then the API will automatically detect the failure and will select another MigratoryData server from the `servers` list. If the client fails to connect to the newly selected server, a status notification MigratoryDataClient.NOTIFY_SERVER_DOWN will be triggered, unless this is modified using MigratoryDataClient.notifyAfterFailedConnectionAttempts()), and a new MigratoryData server of the cluster will be selected again and again until the client will be able to connect to one of the MigratoryData servers of the cluster. When successfully connected, the API will trigger MigratoryData↩ Client.NOTIFY_SERVER_UP.

Furthermore, if the Guaranteed Message Delivery feature is enabled for the MigratoryData cluster, then the messages potentialy published for a subscribed subject during the failover period will be automatically recovered from the cache of the MigratoryData server to which the client reconnects to and a status notification MigratoryData↩ Client.NOTIFY_DATA_SYNC will be triggered for that subject.

If, for example, the failover period is abnormally long, and the client is not able to recover all the messages made available during the failover period for one of its subscribed subjects, then the API will retrieve only the most recent retained message available for that subject and will trigger a MigratoryDataClient.NOTIFY_DATA_RESYNC status notification for that subject, the client behaving as a new client.

For a complete discussion about load balancing, failover, and guaranteed message delivery features see the *Architecture Guide*.

Parameters

servers	an array of strings where each string represents the network address (IP address or DNS domain name and its corresponding port) of a MigratoryData server, optionally prefixed by a weight ranging from `0` to `100`; if the weight prefix is not provided to an address, then the API will automatically assign to that address a default weight `100`

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.7 void MigratoryDataClient.subscribe (array< String > *subjects*)

Subscribe to one or more subjects.

The MigratoryData subjects are strings having a particular syntax. See the Chapter "Concepts" of the Architecture Guide to learn about the syntax of the subjects.

As an example, supposing messages are market data updates having as subjects stock names. Then, to subscribe for the messages having as subjects `/stocks/NYSE/IBM` and `/stocks/Nasdaq/MSFT` use:

```
client.subscribe(array("/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT"));
```

Parameters

subjects	An array of strings representing subjects.

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.8 void MigratoryDataClient.subscribeWithConflation (array< String > *subjects*, int *conflationMillis*)

Subscribe to one or more subjects with conflation.

Subscribe for real-time messages having as subjects the strings provided in the `subjects` parameter.

If the optional parameter `conflationMillis` is used, then for each subject in the `subjects` list given in argument, its messages will be aggregated in the MigratoryData server and published every `conflation↩Millis` milliseconds as aggregated data (containing only the latest value for that subject and its latest field values). The value of `conflationMillis` should be a multiple of `100` milliseconds, otherwise the MigratoryData server will round it to the nearest value multiple of `100` milliseconds (e.g. `76` will be rounded to `0`, `130` will be rounded to `100`, `789` will be rounded to `700`, ...). If the value of `conflationMillis` is `0` (or is rounded to `0`), then no conflation will apply, and data publication will be message-by-message with no message aggregation.

As an example, supposing the messages are market data updates having as subjects stock names. Then, to subscribe for the messages having as subjects `/stocks/NYSE/IBM` and `/stocks/Nasdaq/MSFT` using 1-second conflation the following code will be used:

```
client.subscribeWithConflation(array("/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT"), 1000);
```

The subjects are strings having a particular particular syntax. See the Chapter "Concepts" in the *MigratoryData Architecture Guide* (PDF, HTML) to learn about the syntax of the subjects.

Parameters

subjects	An array of strings representing subjects.
conflationMillis	An optional argument defining the number of milliseconds used to aggregate ("conflate") the messages for each subject in the `subjects` list; default value is `0` meaning that no conflation will apply, and data publication will be message-by-message with no message aggregation.

3.1.2.9 void MigratoryDataClient.subscribeWithHistory (array< String > *subjects*, int *numberOfHistoricalMessages*)

Subscribe to one or more subjects after getting historical messages for those subjects.

The MigratoryData subjects are strings having a particular syntax. See the Chapter "Concepts" of the Architecture Guide to learn about the syntax of the subjects.

Attempt to get the number of historical messages as defined by the argument `numberOfHistorical↩Messages`, for each subject in the argument `subjects`, then subscribe for real-time messages having as subjects the strings provided in the `subjects` parameter.

When Guaranteed Message Delivery is enabled, each MigratoryData server in the cluster maintains an in-memory cache with historical messages for each subject. The cache of each subject is available in all servers of the cluster. The maximum number of messages held in cache is defined by the parameter `MaxCachedMessagesPer↩Subject` of the MigratoryData server which defaults to 1,000 messages. The historical messages are continuously removed from the cache, but it is guaranteed that they are available in the cache at least the number of seconds defined by the parameter `CacheExpireTime` which defaults to 180 seconds.

If the value of `numberOfHistoricalMessages` is higher than the number of historical messages available in the cache, then the client will receive only the messages available in the cache. As a consequence, if you use a value higher than the value of the parameter `MaxCachedMessagesPerSubject` of the MigratoryData server (which defaults to 1000), then you will get the entire cache before subscribing for real-time messages for the subjects specified with the API call.

If the value of `numberOfHistoricalMessages` is `0`, then no historical messages have to be retrieved from the cache and, in this case, this API method is equivalent to the API method MigratoryDataClient.subscribe().

Parameters

subjects	An array of strings representing subjects.
numberOfHistoricalMessages	the number of historical messages to be retrieved from the cache of the server

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.10 void MigratoryDataClient.unsubscribe (array< String > *subjects*)

Unsubscribe from one or more subjects.

Parameters

subjects	subjects to unsubscribe

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.11 void MigratoryDataClient.setEncryption (boolean *encrypted*)

Configure whether to use SSL/TLS encryption when connecting to a MigratoryData server.

When using encryption you should connect to the ports of the MigratoryData server that are configured with the parameter `ListenEncrypted` to listen for encrypted connections.

Parameters

encrypted	indicate whether or not to use an encrypted SSL/TLS connection to communicate with the server

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.

3.1.2.12 void MigratoryDataClient.setEntitlementToken (String *token*)

Assign an entitlement token to the client.

To define which users of your application have access to which subjects, you will first have to configure the parameter `Entitlement`, see the *Configuration Guide*. If you set this parameter on `Custom`, then you can use the *MigratoryData Extension SDK for Entitlement* to build an extension plugin for the MigratoryData server to allow or deny certain users to subscribe to or publish on certain subjects.

Parameters

token	a string representing an entitlement token

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.13 Collection<String> MigratoryDataClient.getSubjects ( )

Return the list of subscribed subjects.

Returns

The list of strings representing the subscribed subjects.

3.1.2.14 void MigratoryDataClient.notifyAfterFailedConnectionAttempts (int *retries*)

Define the number of failed attempts to connect to one or more MigratoryData servers before triggering a status notification MigratoryDataClient.NOTIFY_SERVER_DOWN.

Parameters

retries	the number of the failed attempts to connect to one or more MigratoryData servers before triggering a status notification MigratoryDataClient.NOTIFY_SERVER_DOWN; the default is 1

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.15 void MigratoryDataClient.disconnect ( )

Disconnect from the connected MigratoryData server and dispose the resources used by the connection.

This method should be called when the connection is no longer necessary.

3.1.2.16 void MigratoryDataClient.publish (MigratoryDataMessage *message*)

Publish a message.

If the message includes a closure data, then a status notification will be provided via MigratoryDataListener.on↩Status() to inform whether the message publication has been successful or failed.

Parameters

message	A MigratoryDataMessage message

Exceptions

MigratoryDataException	Exceptions with the error code E_NOT_CONNECTED.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.17 void MigratoryDataClient.setQuickReconnectMaxRetries (int *retries*)

Define the maximum number of retries for the Quick Reconnect fail-over phase.

See MigratoryDataClient.setQuickReconnectInitialDelay() to learn about the Quick Reconnect phase.

Parameters

retries	the maximum number of quick reconnect retries; the default is 3

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.18 void MigratoryDataClient.setQuickReconnectInitialDelay (int *seconds*)

Define the number of seconds to wait before attempting to reconnect to the cluster after a connection failure is detected.

Connection Failure Detection

Connection failure is detected immediately for almost all users. For a few users which are subject to temporary, atypical network conditions, connection failure is detected after 30-40 seconds.

Reconnection Phases and Policies

When a connection failure is detected, the API will attempt to reconnect to the servers of the MigratoryData cluster as follows: First, it will attempt to reconnect up to a number of times as defined by MigratoryDataClient.setQuick↩ ReconnectMaxRetries() using small delays between retries (Quick Reconnection Phase). If the connection cannot be established after the Quick Reconnection Phase, then the API will attempt to reconnect less frequently according to the policy defined by MigratoryDataClient.setReconnectPolicy().

The delays between retries are computed according to the following algorithm where the values of the variables involved are defined by the API methods having substantially the same names:

```
Quick Reconnect Phase (retries <= quickReconnectMaxRetries)
-----------------------------------------------------------

   (retries starts with 1 and increment by 1 at each quick reconnect)

   reconnectDelay = quickReconnectInitialDelay * retries - random(0, quickReconnectInitialDelay)

After Quick Reconnect Phase (retries > quickReconnectMaxRetries)
----------------------------------------------------------------

   (reset retries to start with 1 and increment by 1 at each reconnect)

   If reconnectPolicy is CONSTANT_WINDOW_BACKOFF, then

      reconnectDelay = reconnectTimeInterval

   else if reconnectPolicy is TRUNCATED_EXPONENTIAL_BACKOFF, then

      reconnectDelay = min(reconnectTimeInterval * (2 ^ retries) - random(0, reconnectTimeInterval *
      retries), reconnectMaxDelay)
```

For a few users which are subject to temporary, atypical network conditions, if `reconnectDelay` computed with the algorithm above is less than 10 seconds, then it is rounded to 10 seconds.

Parameters

seconds	The number of seconds to wait before attempting to reconnect to the cluster; default value is `5` seconds.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.

Exceptions

MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.19 void MigratoryDataClient.setReconnectPolicy (String *policy*)

Define the reconnect policy to be used after the Quick Reconnect phase.

See MigratoryDataClient.setQuickReconnectInitialDelay() to learn about the Quick Reconnect phase and the reconnect schedule for the policy defined by this method.

Parameters

policy	the reconnect policy to be used after the Quick Reconnect phase; the possible values are: MigratoryDataClient.CONSTANT_WINDOW_BACKOFF and MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF; the default value is MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF;

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.20 void MigratoryDataClient.setReconnectTimeInterval (int *seconds*)

Define the time interval used for the reconnect schedule after the Quick Reconnect phase.

See MigratoryDataClient.setQuickReconnectInitialDelay() to learn about the Quick Reconnect phase and the reconnect schedule for the policy defined by this method.

Parameters

seconds	A time interval expressed in seconds used for reconnect schedule; default is `20` seconds.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.21 void MigratoryDataClient.setReconnectMaxDelay (int *seconds*)

Define the maximum reconnect delay for the MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF policy.

See MigratoryDataClient.setQuickReconnectInitialDelay() to learn how the value defined by this method is used.

Parameters

seconds	The maximum reconnect delay when the policy MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF is used; default value is 360 seconds.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.22 void MigratoryDataClient.setTransport (String *type*)

Define the transport type used by the client to communicate with the MigratoryData cluster.

Parameters

type	the possible values are: MigratoryDataClient.TRANSPORT_HTTP and MigratoryDataClient.TRANSPORT_WEBSOCKET; the default transport used by the API is MigratoryDataClient.TRANSPORT_WEBSOCKET

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.2.23 void MigratoryDataClient.setConnectionTimeout (long *connectionTimeoutMs*)

Define the timeout value to establish a connection with a MigratoryData server.

Parameters

connectionTimeoutMs	The timeout value (in milliseconds) used by the client to establish a connection with a MigratoryData server; default value is 1000 milliseconds.

Exceptions

MigratoryDataException	Exceptions with the error code E_RUNNING.
MigratoryDataException	Exceptions with the error code E_ILLEGAL_ARGUMENT.

3.1.3 Member Data Documentation

3.1.3.1 final String MigratoryDataClient.CONSTANT_WINDOW_BACKOFF = "CONSTANT_WINDOW_BACKOFF" [static]

A constant used to define the reconnect policy.

See MigratoryDataClient.setQuickReconnectInitialDelay() for more details about this policy.

3.1.3.2 final String MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF = "TRUNCATED_EXPONENTIAL_BACKOFF" [static]

A constant used to define the reconnect policy.

See MigratoryDataClient.setQuickReconnectInitialDelay() for more details about this policy.

3.1.3.3 String MigratoryDataClient.TRANSPORT_HTTP = "TRANSPORT_HTTP" [static]

A constant used to define the transport type.

See MigratoryDataClient.setTransport() for more details about this policy.

3.1.3.4 String MigratoryDataClient.TRANSPORT_WEBSOCKET = "TRANSPORT_WEBSOCKET" [static]

A constant used to define the transport type.

See MigratoryDataClient.setTransport() for more details about this policy.

3.2 MigratoryDataException Class Reference

Define the *error codes* of the API, and their corresponding *error texts* and *error causes*.

Public Member Functions

- int getCode ()
  Get the error code.
- String getCause ()
  Get the error cause.
- String getDetail ()
  Get the error text.

Public Attributes

- int E_INVALID_URL = 1
  Indicates an invalid address for a MigratoryData server.
- int E_RUNNING = 2
  Indicates that the client is already connect to a MigratoryData cluster and the action wanted is not allowed when in this state.
- int E_NOT_CONNECTED = 3
  Indicates that the client is not connect to a MigratoryData cluster and the action wanted is not allowed when in this state. You need to use MigratoryDataClient.connect() .
- int E_ILLEGAL_ARGUMENT = 4
  Indicates that an invalid argument was given to a client method.
- int E_NOT_SET = 5
  Indicates that a required object for the client to connect was not given.

3.2.1 Detailed Description

Define the *error codes* of the API, and their corresponding *error texts* and *error causes*.

3.2.2 Member Function Documentation

3.2.2.1 int MigratoryDataException.getCode ( )

Get the error code.

Returns

The error code.

3.2.2.2 String MigratoryDataException.getCause ( )

Get the error cause.

Returns

The error cause.

3.2.2.3 String MigratoryDataException.getDetail ( )

Get the error text.

Returns

The error text.

3.2.3 Member Data Documentation

3.2.3.1 int MigratoryDataException.E_INVALID_URL = 1

Indicates an invalid address for a MigratoryData server.

The address of a MigratoryData server is incorrectly specified.

3.3 MigratoryDataField Class Reference

Represent a message field.

Public Member Functions

- MigratoryDataField (String name, String value)

  Create a MigratoryDataField object.
- String getName ()

  Get the field name.
- String getValue ()

  Get the field value.

3.3.1 Detailed Description

Represent a message field.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 MigratoryDataField.MigratoryDataField (String *name*, String *value*)

Create a MigratoryDataField object.

Parameters

name	The field name
value	The field value

3.3.3 Member Function Documentation

3.3.3.1 String MigratoryDataField.getName ( )

Get the field name.

Returns

A string representing the field name.

3.3.3.2 String MigratoryDataField.getValue ( )

Get the field value.

Returns

A string representing the field value.

3.4 MigratoryDataListener Interface Reference

The implementation of this interface will handle the messages received from the server for the subscribed subjects as well as various status notifications.

Public Member Functions

- void onMessage (MigratoryDataMessage message)

  This method handles the real-time messages received from a MigratoryData server for the subscribed subjects.
- void onStatus (String type, String info)

  This method handles the status notifications.

3.4.1 Detailed Description

The implementation of this interface will handle the messages received from the server for the subscribed subjects as well as various status notifications.

Use the API method MigratoryDataClient.setListener() to register your listener implementation.

3.4.2 Member Function Documentation

3.4.2.1 void MigratoryDataListener.onMessage (MigratoryDataMessage *message*)

This method handles the real-time messages received from a MigratoryData server for the subscribed subjects.

Parameters

message	An object of type MigratoryDataMessage .

3.4.2.2 void MigratoryDataListener.onStatus (String *type*, String *info*)

This method handles the status notifications.

The possible values of the `status` parameter are:

- `MigratoryDataClient.NOTIFY_SERVER_UP` indicates that the client successfully connected to the MigratoryData server provided in the detail information of the status notification
- `MigratoryDataClient.NOTIFY_SERVER_DOWN` indicates that the client was not able to connect to the MigratoryData server provided in the detail information of the status notification
- `MigratoryDataClient.NOTIFY_DATA_SYNC` indicates that, after a failover reconnection, the client successfully synchronized the subject given in the detail information of the status notification. Moreover, the client recovered all messages made available for that subject during the failover period, if any
- `MigratoryDataClient.NOTIFY_DATA_RESYNC` indicates that, after a failover reconnection, the client successfully
  - synchronized the subject given in the detail information; however, the potential messages made available for that subject during
  - the failover period have not been recovered, the client behaving like a new client which only received the most

– recent retained message available for that subject

- `MigratoryDataClient.NOTIFY_SUBSCRIBE_ALLOW` indicates that the client – identified with the token given in the argument of `MigratoryDataClient.setEntitlementToken()` – is allowed to subscribe to the subject provided in the detail information of the status notification

- `MigratoryDataClient.NOTIFY_SUBSCRIBE_DENY` indicates that the client – identified with the token given in the argument of `MigratoryDataClient.setEntitlementToken()` – is not allowed to subscribe to the subject provided in the detail information of the status notification

- `MigratoryDataClient.NOTIFY_PUBLISH_OK` indicates that the client successfully published the message having the closure data provided in the detail information of the status notification

- `MigratoryDataClient.NOTIFY_PUBLISH_FAILED` indicates that the client was unable to publish the message having the closure data provided in the detail information of the status notification

- `MigratoryDataClient.NOTIFY_PUBLISH_DENIED` indicates that the client was unable to publish the message having the closure data provided in the detail information of the status notification because the client – identified with the token given in the argument of `MigratoryDataClient.setEntitlementToken()` – is not allowed to publish on the subject of the message

Parameters

status	The type of the status notification (see the possible values above).
info	The detail information of the status notification.

3.5 MigratoryDataLogLevel Enum Reference

This class enumerates the MigratoryData logging levels.

Public Attributes

- TRACE

  The `TRACE` level turns on all the logs of the API.
- DEBUG

  The `DEBUG` level turns on the debug, info, warning, and error logs of the API.
- INFO

  The `INFO` level turns on the info, warning, and error logs of the API.
- WARN

  The `WARN` level turns on the warning and error logs of the API.
- ERROR

  The `ERROR` level turns on the error logs of the API.

3.5.1 Detailed Description

This class enumerates the MigratoryData logging levels.

The available logging levels ordered by verbosity are:

- ERROR (less verbose)
- WARN
- INFO
- DEBUG
- TRACE (most verbose)

For production usage, we recommend the default `INFO` logging level.

3.6 MigratoryDataLogListener Interface Reference

The implementation of this interface will handle the log messages produced by the library.

Public Member Functions

- void onLog (String log, MigratoryDataLogLevel level)
  This method handles the log messages outputted by the library.

3.6.1 Detailed Description

The implementation of this interface will handle the log messages produced by the library.

Use the API method MigratoryDataClient.setLogListener() to register your log listener implementation.

3.6.2 Member Function Documentation

3.6.2.1 void MigratoryDataLogListener.onLog (String *log,* MigratoryDataLogLevel *level*)

This method handles the log messages outputted by the library.

Parameters

log	the log message outputted by the library
level	the verbosity level of the log message

3.7 MigratoryDataMessage Class Reference

Represent a message.

Public Member Functions

- MigratoryDataMessage (String subject, String content, String closure, array< MigratoryDataField > fields, String replySubject)

  Create a MigratoryDataMessage object.
- String getSubject ()

  Get the subject of the message.
- String getContent ()

  Get the content of the message.
- array< MigratoryDataField > getFields ()

  Get the fields of the message.
- array< String, String > getFieldsAsMap ()

  Get the fields of the message.
- String getClosure ()

  Get the closure of the message.
- boolean isSnapshot ()

  Test whether the message is a snapshot message or not.
- void setReplyToSubject (String subject)

  Set the subject to be used to reply to this message.
- String getReplyToSubject ()

  Get the subject to be used to reply to this message.
- String toString ()

  Return a string representation of the message.

3.7.1 Detailed Description

Represent a message.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 MigratoryDataMessage.MigratoryDataMessage (String *subject*, String *content*, String *closure*, array< MigratoryDataField > *fields*, String *replySubject*)

Create a MigratoryDataMessage object.

Parameters

subject	The subject of the message
content	The content of the message
closure	The closure of the message (OPTIONAL)
fields	The fields of the message (OPTIONAL)
replySubject	The reply subject of the message (OPTIONAL)

3.7.3 Member Function Documentation

3.7.3.1 String MigratoryDataMessage.getSubject ( )

Get the subject of the message.

Returns

A string representing the subject of the message

3.7.3.2 String MigratoryDataMessage.getContent ( )

Get the content of the message.

Returns

A string representing the content of the message

3.7.3.3 array<MigratoryDataField> MigratoryDataMessage.getFields ( )

Get the fields of the message.

Returns

The fields of the message as a list of MigratoryDataField objects

3.7.3.4 array<String, String> MigratoryDataMessage.getFieldsAsMap ( )

Get the fields of the message.

Returns

The fields of the message as a map.

3.7.3.5 String MigratoryDataMessage.getClosure ( )

Get the closure of the message.

Returns

The closure data of the message

3.7.3.6 boolean MigratoryDataMessage.isSnapshot ( )

Test whether the message is a snapshot message or not.

Returns

true if the message is a snapshot message

3.7.3.7 void MigratoryDataMessage.setReplyToSubject (String *subject*)

Set the subject to be used to reply to this message.

If a reply subject is attached to a message with this method, the message acts as a request. The clients which receive a request message will be able to reply by sending a message having as subject the reply subject.

If the reply subject is not already subscribed, it is subscribed by the API library implicitly. It can be reused for subsequent request/reply interactions (and even for receiving multiple replies to one request). When it is not needed anymore, it should be unsubscribed explicitly.

Parameters

subject	The subject to be used to reply to this message.

3.7.3.8 String MigratoryDataMessage.getReplyToSubject ( )

Get the subject to be used to reply to this message.

A client which receives a message containing a reply subject should interpret the message as a request. It has the option to use the reply subject - extracted from the message with this method - to send a reply.

Returns

The subject to be used to reply to this message.

Index