MigratoryData Client API for NodeJS

Developer's Guide and Reference Manual

August 23, 2019

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 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 Node.Js clients for MigratoryData Server

A typical API usage is as follows:

1.2.1 Step 1 - Include the library

For using this API please reference in your web application the file `migratorydata-client.js` which is located in the folder `lib` of this API package.

1.2.2 Step 2 - Specify the cluster of MigratoryData servers where to connect to

Use the API method MigratoryDataClient.setServers() to specify one or more MigratoryData servers to which your Node.js client will connect to. In fact, the Node.js client will connect to only one of the MigratoryData servers in this list. But, defining two or more MigratoryData servers is recommended to achieve fail-over. Supposing the MigratoryData server to which the Node.js client connected goes down, then the API will automatically reconnect to another MigratoryData server in the list.

If you define two or more MigratoryData servers, then all the MigratoryData servers should provide the same level of data redundancy, by making available for subscription the same set of subjects.

1.2.3 Step 3 - Define the handler function for real-time messages

Use the API method MigratoryDataClient.setMessageHandler() to define the *message handler*, a function defined by your web application that will handle the real-time messages received from a MigratoryData server. The *message handler* must have the following signature:

```
function <messageHandlerFunction>(Object messages);
```

where `<messageHandlerFunction>` can be any function name of your choice. Its `messages` argument is an array of messages, where each message in the array is an object having the following properties:

- `subject` - the subject of the message
- `content` - the content of the message
- `fields` - an array of fields where each field is an object with two properties: `name` (the name of the field) and `value` (the value of the field)
- `isSnapshot` - indicate whether the message is an initial snapshot message or not

1.2.4 Step 4 - Define the handler function for status notifications

Use the API method MigratoryDataClient.setStatusHandler() to define the *status handler*, a function defined by your web application that will handle the status notifications. The *status handler* must have the following signature:

```
function <statusHandlerFunction>(Object status);
```

where `<statusHandlerFunction>` can be any function name of your choice. Its `status` argument is an object having two properties:

- `type` - the type of the status notification
- `info` - the detail information of the status notification

1.2.5 Step 5 - 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.

1.2.6 Step 6 - Handle the real-time messages and status notifications

Handle the received messages in your *message handler* defined above, and, optionally, handle the status notifications in your *status handler* defined above (you may choose to ignore the status notifications by not defining a *status handler*).

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 subjects, getting real-time messages for the subscribed subjects, and publishing real-time messages.

Public Member Functions

- void setServers (string[ ] servers, boolean connect)

  Specify a cluster of one or more MigratoryData servers to which the client will connect to.
- void setMessageHandler (function messageHandler)

  Specify a custom function name used to process the real-time messages received from a MigratoryData server.
- void setStatusHandler (function statusHandler)

  Specify a custom function name used to process the status notifications.
- void setEntitlementToken (string token)

  Assign an authorization token to the client.
- void subscribe (string[ ] subjects)

  Subscribe to one or more subjects.
- void subscribeWithHistory (string[ ] subjects, int numberOfHistoricalMessages)

  Subscribe to one or more subjects after getting historical messages for those subjects.
- void subscribeWithConflation (string[ ] subjects, int conflationMillis)

  Subscribe to one or more subjects with conflation.
- void unsubscribe (string[ ] subjects)

  Unsubscribe from one or more subjects.
- void publish (Object message)

  Publish a message.
- string[ ] getSubjects ()

  Return the list of subscribed subjects.
- void notifyAfterReconnectRetries (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 setQuickReconnectInitialDelay (int seconds)

  Define the number of seconds to wait before attempting to reconnect to the cluster after a connection failure is detected.
- void setQuickReconnectMaxRetries (int retries)

Define the maximum number of retries for the Quick Reconnect failover phase.

- 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 disconnect ()

  Disconnect from the connected MigratoryData server and dispose the resources used by the connection.
- string getInfo ()

  Return various statistical information.

Public Attributes

- string NOTIFY_SERVER_UP

  Indicate that the client successfully connected to a MigratoryData server.
- string NOTIFY_SERVER_DOWN

  Indicate that the client failed to connect to a MigratoryData server.
- string NOTIFY_DATA_SYNC

  After a failover reconnection, the client synchronized a subscribed subject with the latest message available for that subject, as well as with all messages published during the failover for that subject.
- string NOTIFY_DATA_RESYNC

  After a failover reconnection, the client synchronized a subscribed subject with the latest message available for that subject, but not with the potential messages published during the failover, therefore behaving as a new client.
- string NOTIFY_SUBSCRIBE_ALLOW

  Indicate that the client was authorized to subscribe to a subject.
- string NOTIFY_SUBSCRIBE_DENY

  Indicate that the client was not authorized to subscribe to a subject.
- string NOTIFY_PUBLISH_OK

  Indicate that the client successfully published a message.
- string NOTIFY_PUBLISH_FAILED

  Indicate that the client was unable to publish a message.
- string NOTIFY_PUBLISH_DENIED

  Indicate that the client was unable to publish a message because it is not allowed by the entitlement rules you defined.
- string NOTIFY_PUBLISH_NO_SUBSCRIBER

  Indicate that the client was unable to publish a message because there is no client subscribed to the subject of the message.
- string NOTIFY_UNSUPPORTED_BROWSER

  Indicate that the client runs in a browser which is not supported by the API.
- string CONSTANT_WINDOW_BACKOFF

  A constant used to define the reconnect policy.
- string TRUNCATED_EXPONENTIAL_BACKOFF

  A constant used to define the reconnect policy.

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 subjects, getting real-time messages for the subscribed subjects, and publishing real-time messages.

> In this reference manual, the following type notations are used: `int`, `string`, `bool`, `void`, and the notation `string[]` (indicating an array of strings) to better characterize the API. JavaScript is a dynamic language that doesn't have declared types, the type notations used in this document are only included to enhance the documentation.

3.1.2 Member Function Documentation

3.1.2.1 void MigratoryDataClient.setServers (string[ ] *servers,* boolean *connect*)

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

If you specify two or more MigratoryData servers, then all these MigratoryData servers should provide the same level of data redundancy, by making available for subscription the same set of subjects. This is required for achieving (weighted) load balancing, failover, and guaranteed message delivery of the system. In this way, the MigratoryData servers of the `servers` list form a *cluster*.

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:

```
var servers = new Array("http://p1.example.com:80", "http://p2.example.com:80");
MigratoryDataClient.setServers(servers);
```

or, given the fact that the standard HTTP port `80` is used by default for URLs and using the JavaScript square bracket notation for arrays, a more concise JavaScript code can be used:

```
MigratoryDataClient.setServers(["http://p1.example.com", "http://p2.example.com"]);
```

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 in 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:
>
> ```
> MigratoryDataClient.setServers(["100 http://p1.example.com", "50 http://p2.example.com"]);
> ```
>
> 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 new selected server, a status notification MigratoryDataClient.NOTIFY_SERVER_DOWN will be triggered (unless you modify the number of failed attempts with MigratoryDataClient.notifyAfterReconnectRetries()), and a new MigratoryData server in the cluster will be selected again and again until the client will be able to connect to one of the MigratoryData servers in the cluster. When successfully connected, the API will trigger a status notification MigratoryDataClient.NOTIFY_SERVER_UP.

> Furthermore, if guaranteed message delivery is enabled, then the potential messages published for a subscribed subject during the failover period, will be automatically retrieved from the cache of the MigratoryData server to which the client reconnects to and a status notification MigratoryDataClient.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 retrieve, after a failover reconnection, the messages published during the failover period for one of its subscribed subjects, then the API will retrieve only the most recent 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 which connects to the cluster at the moment of the failover reconnection.

For a complete discussion related to load balancing, failover, and guaranteed message delivery features see the *MigratoryData Architecture Guide* (PDF, HTML).

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`.
connect	An optional parameter which defaults to `false`. In order to optimize the communication with the server (especially for old browsers), the library can postpone connecting to the server until the first subscribe or publish operation is performed, and therefore NOTIFY_SERVER_UP is not received until the first subscribe or publish operation is performed. In order to connect to the server immediately, without needing to subscribe or publish, set this optional paramter on `true`.

3.1.2.2 void MigratoryDataClient.setMessageHandler (function *messageHandler*)

Specify a custom function name used to process the real-time messages received from a MigratoryData server.

This API call is used to define the *message handler* which is a function defined by your application that will handle the real-time messages received from a MigratoryData server. Your message handler must have the following signature:

`function <messageHandlerFunction>(Object messages);`

where *<messageHandlerFunction>* can be any function name of your choice. Its `messages` argument is an array of messages, where each message in the array is an object having the following properties:

- `subject` - the subject of the message
- `content` - the content of the message
- `fields` - an array of fields where each field is an object with two properties: `name` (the name of the field) and `value` (the value of the field)
- `isSnapshot` - indicate whether the message is an initial snapshot message or not
- `replyToSubject` - the subject used to reply to this message

This is a code example:

```
MigratoryDataClient.setMessageHandler(messageHandler);


function messageHandler(messages) {
   var out = "Got message(s): [ ";
   for (var i = 0; i < messages.length; i++) {
      out += messages[i].subject + " = " + messages[i].content + ", fields = [";
      for (var j = 0; j < messages[i].fields.length; j++) {
         out += messages[i].fields[j].name + " = " + messages[i].fields[j].value + " ";
      }
      out += "]";
   }
   out += " ]";
   alert(out);
}
```

Parameters

messageHandler	The name of a custom function used to handle the real-time messages.

3.1.2.3 void MigratoryDataClient.setStatusHandler (function *statusHandler*)

Specify a custom function name used to process the status notifications.

This API call is used to define the *status handler* which is a function defined by your application that will handle the status notifications. Your status handler must have the following signature:

```
function <statusHandlerFunction>(Object status);
```

where `<statusHandlerFunction>` can be any function name of your choice. Its `status` argument is an object having two properties:

- `type` - the type of the status notification
- `info` - the detail information of the status notification

The possible values for the type of the status notifications 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 received the messages published during the failover period for this subject.
- `MigratoryDataClient.NOTIFY_DATA_RESYNC` indicates that, after a failover reconnection, the client successfully synchronized the subject given in the detail information of the status notification. However, the client have not received the potential messages published during the failover period for this subject, the client behaving like a new client which just connected to the MigratoryData server.
- `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

- MigratoryDataClient.NOTIFY_PUBLISH_NO_SUBSCRIBER indicates that the client was unable to publish the message having the closure data provided in the detail information of the status notification because there is no client subscribed to the subject of the message

This is a code example:

```
MigratoryDataClient.setStatusHandler(statusHandler);


function statusHandler(status) {
   alert("Got status notification, type = " + status.type + ", info = " + status.info);
}
```

Parameters

statusHandler	The name of a custom function used to handle the status notifications.

3.1.2.4 void MigratoryDataClient.setEntitlementToken (string *token*)

Assign an authorization token to the client.

To define which users of your application have access to which subjects, you will first have to set the parameter Entitlement on true in the configuration file of the MigratoryData server — see the parameter Entitlement in the *MigratoryData Configuration Guide* (PDF, HTML).

Then, you will have to use the entitlement-related part of the MigratoryData Extension API to allow or deny certain users to subscribe / publish to certain subjects.

Parameters

token	A string representing an authorization token.

3.1.2.5 void MigratoryDataClient.subscribe (string[ ] *subjects*)

Subscribe to one or more subjects.

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

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 the following code will be used:

```
var subjects = new Array("/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT");
MigratoryDataClient.subscribe(subjects);
```

or more simple, using the JavaScript square bracket notation for arrays:

```
MigratoryDataClient.subscribe(["/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT"]);
```

The subjects are strings having a 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.

3.1.2.6 void MigratoryDataClient.subscribeWithHistory (string[ ] *subjects*, int *numberOfHistoricalMessages*)

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

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

When Guranteed 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 `MaxCachedMessagesPerSubject` 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 then 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.

```
var subjects = new Array("/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT");
MigratoryDataClient.subscribeWithHistory(subjects, 10);
```

or more simple, using the JavaScript square bracket notation for arrays:

```
MigratoryDataClient.subscribeWithHistory(["/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT"], 10);
```

The subjects are strings having a 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.
numberOfHistoricalMessages	The number of historical messages to be retrieved from the cache of the MigratoryData server. A value `0` means that no historical messages has to be retrieved and, in this case, this API method is equivalent to the API method MigratoryDataClient.subscribe(). A value larger that the value of the parameter `MaxCachedMessagesPerSubject` means the entire cache is retrieved.

3.1.2.7 void MigratoryDataClient.subscribeWithConflation (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:

```
var subjects = new Array("/stocks/NYSE/IBM", "/stocks/Nasdaq/MSFT");
MigratoryDataClient.subscribeWithConflation(subjects, 1000);
```

or more simple, using the JavaScript square bracket notation for arrays:

```
MigratoryDataClient.subscribeWithConflation(["/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.

3.1.2.8 void MigratoryDataClient.unsubscribe (string[ ] *subjects*)

Unsubscribe from one or more subjects.

Unsubscribe from the subscribed subjects provided in the `subjects` parameter.

Parameters

subjects	An array of strings representing subjects.

3.1.2.9 void MigratoryDataClient.publish (Object *message*)

Publish a message.

The message format is as follows:

```
{ subject: "some-subject",
  content: "some-content",
  fields: [
     {field-name-1: "field-value-1"},
     {field-name-2: "field-value-2"},
     ...
  ],
  closure: "some-message-id"
}
```

If the message includes a replyToSubject, then it acts as a request. The clients which receive the message, will be able to reply by sending a message having as subject the reply subject defined by replyToSubject.

If the message includes a closure, then a status notification will be provided to inform whether the message publication has been successful or failed.

Parameters

message	A JavaScript object having the structure defined above.

3.1.2.10 string [ ] MigratoryDataClient.getSubjects ( )

Return the list of subscribed subjects.

Returns

An array of strings representing the subscribed subjects.

3.1.2.11 void MigratoryDataClient.notifyAfterReconnectRetries (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; default value is `1`.

3.1.2.12 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 running modern browsers. For a few users running modern browsers (and being subject to temporary, atypical network conditions) as well as for all users running older browsers without WebSocket support, 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 acoording to the following algoritm 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, reconnectTimeInter
```

For a few users running modern browsers (and being subject to temporary, atypical network conditions) as well as for all users running older browsers without WebSocket support, 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.

3.1.2.13 void MigratoryDataClient.setQuickReconnectMaxRetries (int *retries*)

Define the maximum number of retries for the Quick Reconnect failover phase.

Parameters

retries	The maximum number of quick reconnect retries; default value is `3`.

3.1.2.14 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.

3.1.2.15 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 how the value defined by this API method is used for the reconnect schedule.

Parameters

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

3.1.2.16 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 API method is used.

Parameters

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

3.1.2.17 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.

3.1.2.18 string MigratoryDataClient.getInfo ( )

Return various statistical information.

Returns

A string with various statistical information useful for debugging or logging.

3.1.3 Member Data Documentation

3.1.3.1 string MigratoryDataClient.NOTIFY_SERVER_UP

Indicate that the client successfully connected to a MigratoryData server.

This constant indicates that the client successfully connected to one of the MigratoryData servers defined with the API method MigratoryDataClient.setServers().

3.1.3.2 string MigratoryDataClient.NOTIFY_SERVER_DOWN

Indicate that the client failed to connect to a MigratoryData server.

This constant indicates that the client failed to connect to one of the MigratoryData servers defined with the API method MigratoryDataClient.setServers().

3.1.3.3 string MigratoryDataClient.NOTIFY_DATA_SYNC

After a failover reconnection, the client synchronized a subscribed subject with the latest message available for that subject, as well as with all messages published during the failover for that subject.

This constant indicates that the client successfully synchronized the subject provided in the detail information of the status notification. Also, the potential messages published for that subject during the failover period have been successfully retrieved at the moment of the reconnection.

3.1.3.4 string MigratoryDataClient.NOTIFY_DATA_RESYNC

After a failover reconnection, the client synchronized a subscribed subject with the latest message available for that subject, but not with the potential messages published during the failover, therefore behaving as a new client.

This constant indicates that the client successfully synchronized the subject provided in the detail information of the status notification. However, the client was unable to get the messages published during the failover. Therefore, it behaves like a new client which connects to the MigratoryData server at the moment of the failover reconnection.

3.1.3.5 string MigratoryDataClient.NOTIFY_SUBSCRIBE_ALLOW

Indicate that the client was authorized to subscribe to a subject.

This constant indicates that the client – identified with the token defined with the API method MigratoryData↩Client.setEntitlementToken() – is allowed to subscribe to the subject provided in the detail information of the status notification.

3.1.3.6 string MigratoryDataClient.NOTIFY_SUBSCRIBE_DENY

Indicate that the client was not authorized to subscribe to a subject.

This constant indicates that the client – identified with the token defined with the API method MigratoryData↩Client.setEntitlementToken() – is not allowed to subscribe to the subject provided in the detail information of the status notification.

3.1.3.7 string MigratoryDataClient.NOTIFY_PUBLISH_OK

Indicate that the client successfully published a message.

This constant is used to indicate that the publication of the message, having the closure provided in the detail information of the status notification, has succeeded.

3.1.3.8 string MigratoryDataClient.NOTIFY_PUBLISH_FAILED

Indicate that the client was unable to publish a message.

This constant is used to indicate that the publication of the message, having the closure provided in the detail information of the status notification, has failed.

3.1.3.9 string MigratoryDataClient.NOTIFY_PUBLISH_DENIED

Indicate that the client was unable to publish a message because it is not allowed by the entitlement rules you defined.

This constant is used to indicate that the publication of the message, having the closure provided in the detail information of the status notification, has failed. The publication failed because the client – identified with the token defined with the API method MigratoryDataClient.setEntitlementToken() – is not allowed to publish on the subject of the message.

3.1.3.10 string MigratoryDataClient.NOTIFY_PUBLISH_NO_SUBSCRIBER

Indicate that the client was unable to publish a message because there is no client subscribed to the subject of the message.

This constant is used to indicate that the publication of the message, having the closure provided in the detail information of the status notification, has failed. The publication failed because there is no client then subscribed to the subject of the message.

3.1.3.11 string MigratoryDataClient.NOTIFY_UNSUPPORTED_BROWSER

Indicate that the client runs in a browser which is not supported by the API.

This constant indicates that the client tries to run in a browser which is not supported by the API.

The API has support for all standard browsers for desktops and mobile devices. It has support for all recent versions of these standard browsers as well as for older versions — as old as Internet Explorer version 6 (released in 2001). However browsers older that IE6 or non-standard browsers might not be supported by the API.

3.1.3.12 string MigratoryDataClient.CONSTANT_WINDOW_BACKOFF

A constant used to define the reconnect policy.

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

3.1.3.13 string MigratoryDataClient.TRUNCATED_EXPONENTIAL_BACKOFF

A constant used to define the reconnect policy.

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

Index