MigratoryData Client API for iOS

Developer's Guide and Reference Manual

August 8, 2019

Contents

Chapter 1

Developer's Guide

This guide includes the following sections:

- Overview
- Creating iOS clients for MigratoryData Server
- Examples

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 iOS clients for MigratoryData Server

A typical API usage is as follows:

1.2.1 Step 1 - Include the library.

Include the headers of the API located in the folder `lib` of this API package.

Add the `lib` folder to the *Include Directories* of your iOS application.

The API library is available in the folder `lib` of this API package. Add the `lib` folder to the *Library Directories* of your iOS application. Also, add the API library itself to the list of *Library Dependencies* of your iOS application.

1.2.2 Step 2 - Define the listener class to get the real-time messages and status notifications.

The listener should implement the MigratoryDataListener interface.

Use the API call MigratoryDataClient::setListener: to attach your listener implementation.

1.2.3 Step 3 - Specify the list of MigratoryData servers where to connect to.

Use the API method MigratoryDataClient::setServers: to specify one or more MigratoryData servers to which your iOS client will connect to. In fact, the iOS 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 (and horizontal scalling / load balancing). Supposing the MigratoryData server - to which the iOS client connected - goes down, then the API will automatically reconnect that client to another MigratoryData server in the list.

1.2.4 Step 4 Subscribe to subjects and publish messages

Use the API method MigratoryDataClient::subscribe: to subscribe to subjects and use the API method MigratoryDataClient::publish: to publish messages.

1.2.5 Step 5 - 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.

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 compile and run them.

Chapter 2

Deprecated List

Member NOTIFY_PUBLISH_NO_SUBSCRIBER

no more is use.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

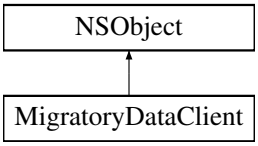
Chapter 6

Class Documentation

6.1 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.

Inheritance diagram for MigratoryDataClient:



Instance Methods

- (id) - init

  Create a MigratoryDataClient object.
- (void) - setLogType:

  Configure the logging level.
- (void) - setListener:

  Attach a MigratoryDataListener for handling real-time messages and status notifications.
- (void) - setServers:

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

  Subscribe to one or more subjects.
- (void) - subscribeWithConflation:conflationTimeMillis:

  Subscribe to one or more subjects with conflation.
- (void) - unsubscribe:

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

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

  Assign an authorization token to the client.
- (NSArray ∗) - getSubjects

Return the list of subscribed subjects.

- (void) - setServersDownBeforeNotify:

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

- (void) - publish:

  Publish a message.

- (void) - pause

  Pause the API operation.

- (void) - resume

  Resume the API operation.

- (void) - dispose

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

- (void) - setQuickReconnectMaxRetries:

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

- (void) - setQuickReconnectInitialDelay:

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

- (void) - setReconnectPolicy:

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

- (void) - setReconnectTimeInterval:

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

- (void) - setReconnectMaxDelay:

  Define the maximum reconnect delay for the TRUNCATED_EXPONENTIAL_BACKOFF policy.

- (void) - setExternalToken:

  Assign an external token to a client.

- (void) - **setNumberOfHistoricalMessages:**

6.1.1 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.

6.1.2 Method Documentation

6.1.2.1 setLogType:()

```
- (void) setLogType:
            (LogType) logLevel
```

Configure the logging level.

It is advisable to configure this first if you want to log as much as possible. The default log level is LogType.LOG_↩ INFO.

Parameters

logLevel	The logging verbosity (LogType.LOG_ERROR, LogType.LOG_INFO, LogType.LOG_DEBUG or LogType.LOG_TRACE); by default LogType.LOG_INFO is configured.

6.1.2.2 setListener:()

```
- (void) setListener:
            (NSObject< MigratoryDataListener > *) listener
```

Attach a MigratoryDataListener for handling real-time messages and status notifications.

Parameters

listener	An instance of a class which implements the MigratoryDataListener interface

6.1.2.3 setServers:()

```
- (void) setServers:
            (NSArray *) servers
```

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:

```
NSArray *servers = [NSArray arrayWithObjects:"p1.example.com:80", @"p2.example.com:80"];
[client setServers:servers];
```

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:

```
NSArray *servers = [NSArray arrayWithObjects:"100 p1.example.com", @"50 p2.example.com"];
[client setServers:servers];
```

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 NOTIFY_SERVER_DOWN will be triggered (unless you modify the number of failed attempts with MigratoryDataClient::setServersDownBeforeNotify:), 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 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 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 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`.

6.1.2.4 subscribe:()

```
- (void) subscribe:
             (NSArray *) 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:

```
NSMutableArray *subjects = [NSMutableArray new];
[subjects addObject:"/stocks/NYSE/IBM"];
[subjects addObject:@"/stocks/Nasdaq/MSFT"];
[client subscribe:subjects];
```

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.

6.1.2.5 subscribeWithConflation:conflationTimeMillis:()

```
- (void) subscribeWithConflation:
             (NSArray *) subjects
             conflationTimeMillis:(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:

```
NSMutableArray *subjects = [NSMutableArray new];
[subjects addObject:"/stocks/NYSE/IBM"];
[subjects addObject:@"/stocks/Nasdaq/MSFT"];
[client subscribeWithConflation:subjects conflationTimeMillis: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.

6.1.2.6 unsubscribe:()

```
- (void) unsubscribe:
             (NSArray *) 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.

6.1.2.7 setEncryption:()

```
- (void) setEncryption:
             (BOOL) encryption
```

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

When using encryption you have to connect to the ports of the MigratoryData servers that are configured to listen for encrypted connections. See the parameter `ListenEncrypted` in the *MigratoryData Configuration Guide* (PDF, HTML).

Parameters

encryption	Determine whether the client connects to the MigratoryData server using an encrypted SSL/TLS connection

6.1.2.8 setEntitlementToken:()

```
- (void) setEntitlementToken:
             (NSString *) 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.

6.1.2.9 getSubjects()

```
- (NSArray *) getSubjects
```

Return the list of subscribed subjects.

Returns

The list of strings representing the subscribed subjects.

6.1.2.10 setServersDownBeforeNotify:()

```
- (void) setServersDownBeforeNotify:
            (int) n
```

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

Parameters

n	The number of the failed attempts to connect to one or more MigratoryData servers before triggering a status notification NOTIFY_SERVER_DOWN; default value is `1`.

6.1.2.11 publish:()

```
- (void) publish:
            (MigratoryDataMessage *) message
```

Publish a message.

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

Parameters

message	A MigratoryDataMessage message

6.1.2.12 pause()

```
- (void) pause
```

Pause the API operation.

If the application built with this API library enters into background, then it is recommended to use this API call. It will disconnect the user from the MigratoryData sever but will preserve the user's context (including the cluster definition and subscribed subjects) in order to reconnect to the cluster later when the API method MigratoryData↩ Client::resume is called.

Moreover, if the cluster is configured with guaranteed message delivery, then when the user will reconnect to the cluster using MigratoryDataClient::resume, it will get all messages published since the MigratoryDataClient::pause method was called, provided however that the duration between the time when MigratoryDataClient::pause method was called and the MigratoryDataClient::resume method was called is less than the value defined by the parameter `CacheExpireTime` of the MigratoryData server (see details about this parameter in *MigratoryData Configuration Guide* (PDF, HTML).

6.1.2.13 resume()

```
- (void) resume
```

Resume the API operation.

If the application was paused with the MigratoryDataClient::pause method, then this API call will attempt to reconnect the user to the cluster. Also, if guaranteed message delivery is enabled, this method also retrieves all messages published since the MigratoryDataClient::pause has been called provided that the duration between the time when MigratoryDataClient::pause method was called and the MigratoryDataClient::resume method was called is smaller than the value defined by the parameter `CacheExpireTime`.

This method will be typically used when the application switches to foreground.

6.1.2.14 dispose()

```
- (void) dispose
```

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.

6.1.2.15 setQuickReconnectMaxRetries:()

```
- (void) setQuickReconnectMaxRetries:
             (int) retries
```

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

Parameters

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

6.1.2.16 setQuickReconnectInitialDelay:()

```
- (void) 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 (the 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.

6.1.2.17 setReconnectPolicy:()

```
- (void) setReconnectPolicy:
            (NSString *) 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 CONSTANT_WINDOW_BACKOFF and TRUNCATED_EXPONENTIAL_BACKOFF; the default value is TRUNCATED_EXPONENTIAL_BACKOFF.

6.1.2.18 setReconnectTimeInterval:()

```
- (void) 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.

Parameters

seconds	A time interval expressed in seconds; default is `20` seconds.

6.1.2.19 setReconnectMaxDelay:()

```
- (void) setReconnectMaxDelay:
            (int) seconds
```

Define the maximum reconnect delay for the 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 TRUNCATED_EXPONENTIAL_BACKOFF is used; default value is 360 seconds.

6.1.2.20 setExternalToken:()

```
- (void) setExternalToken:
            (NSString *) externalToken
```

Assign an external token to a client.

An external token which is provided by a client using this method is typically used by a MigratoryData plugin to enable that client to communicate with an external service.

For example the MigratoryData plugin for Firebase needs an FCM token in order to be able to push notifications via the Firebase service to a mobile client. The mobile client can provide the FCM token to the plugin using this method.

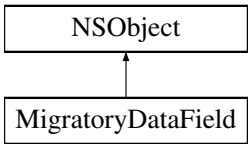
Parameters

externalToken	A string representing an external token

6.2 MigratoryDataField Class Reference

Represent a message field.

Inheritance diagram for MigratoryDataField:



Instance Methods

- (id) - init:value:

  Create a MigratoryDataField object.
- (NSString *) - getName

  Get the field name.
- (NSString *) - getValue

  Get the field value.

6.2.1 Detailed Description

Represent a message field.

6.2.2 Method Documentation

6.2.2.1 init:value:()

```
- (id) init:
             (NSString *) name
             value:(NSString *) value
```

Create a MigratoryDataField object.

Parameters

name	The field name
value	The field value

6.2.2.2 getName()

```
- (NSString *) getName
```

Get the field name.

Returns

A string representing the field name.

6.2.2.3 getValue()

```
- (NSString *) getValue
```

Get the field value.

Returns

A string representing the field value.

6.3 <MigratoryDataListener> Protocol Reference

Implementations of this interface can handle the real-time messages received for the subscribed subjects as well as various status notifications.

Instance Methods

- (void) - onMessage:

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

  This method handles the status notifications.

6.3.1 Detailed Description

Implementations of this interface can handle the real-time messages received for the subscribed subjects as well as various status notifications.

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

6.3.2 Method Documentation

6.3.2.1 onMessage:()

```
- (void) 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 .

6.3.2.2 onStatus:info:()

```
- (void) onStatus:
            (NSString *) status
            info:(NSString *) info
```

This method handles the status notifications.

The possible values of the `status` parameter are:

- NOTIFY_SERVER_UP indicates that the client successfully connected to the MigratoryData server provided in the detail information of the status notification

- 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

- 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.

- 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.

- 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

- 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

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

- 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

- 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

- 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

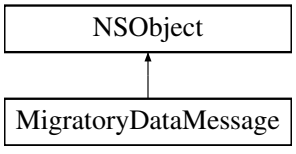
Parameters

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

6.4 MigratoryDataMessage Class Reference

Represent a message.

Inheritance diagram for MigratoryDataMessage:



Instance Methods

- (id) - init:content:

  Create a MigratoryDataMessage object.
- (id) - init:content:fields:

  Create a MigratoryDataMessage object.
- (id) - init:content:closure:

  Create a MigratoryDataMessage object.
- (id) - init:content:fields:closure:

  Create a MigratoryDataMessage object.
- (NSString ∗) - getSubject

  Get the subject of the message.
- (NSString ∗) - getContent

  Get the content of the message.
- (NSArray ∗) - getFields

  Get the fields of the message.
- (NSString ∗) - getClosure

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

  Test whether the message is a snapshot message or not.
- (NSString ∗) - getReplyToSubject
- (void) - setReplyToSubject:

6.4.1 Detailed Description

Represent a message.

6.4.2 Method Documentation

6.4.2.1 init:content:()

```
- (id) init:
             (NSString *) subject
             content:(NSString *) content
```

Create a MigratoryDataMessage object.

Parameters

subject	The subject of the message
content	The content of the message

6.4.2.2 init:content:fields:()

```
- (id) init:
              (NSString *) subject
             content:(NSString *) content
             fields:(NSArray *) fields
```

Create a MigratoryDataMessage object.

Parameters

subject	The subject of the message
content	The content of the message
fields	The fields of the message

6.4.2.3 init:content:closure:()

```
- (id) init:
              (NSString *) subject
             content:(NSString *) content
             closure:(NSString *) closure
```

Create a MigratoryDataMessage object.

Parameters

subject	The subject of the message
content	The content of the message
closure	The closure of the message

6.4.2.4 init:content:fields:closure:()

```
- (id) init:
              (NSString *) subject
             content:(NSString *) content
             fields:(NSArray *) fields
             closure:(NSString *) closure
```

Create a MigratoryDataMessage object.

Parameters

subject	The subject of the message
content	The content of the message
fields	The fields of the message
closure	The closure of the message

6.4.2.5 getSubject()

```
- (NSString *) getSubject
```

Get the subject of the message.

Returns

A string representing the subject of the message

6.4.2.6 getContent()

```
- (NSString *) getContent
```

Get the content of the message.

Returns

A string representing the content of the message

6.4.2.7 getFields()

```
- (NSArray *) getFields
```

Get the fields of the message.

Returns

The fields of the message as a list of MigratoryDataField objects

6.4.2.8 getClosure()

```
- (NSString *) getClosure
```

Get the closure of the message.

Returns

The closure data of the message

6.4.2.9 isSnapshot()

```
- (bool) isSnapshot
```

Test whether the message is a snapshot message or not.

Returns

true if the message is a snapshot message

6.4.2.10 getReplyToSubject()

```
- (NSString *) 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.

6.4.2.11 setReplyToSubject:()

```
- (void) setReplyToSubject:
             (NSString *) replyToSubject
```

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.

Chapter 7

File Documentation

7.1 src/MigratoryDataClient.h File Reference

Include the declaration of the MigratoryDataClient class.

Classes

- class MigratoryDataClient

  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.

7.1.1 Detailed Description

Include the declaration of the MigratoryDataClient class.

7.2 src/MigratoryDataField.h File Reference

Include the declaration of the MigratoryDataField class.

Classes

- class MigratoryDataField

  Represent a message field.

7.2.1 Detailed Description

Include the declaration of the MigratoryDataField class.

7.3 src/MigratoryDataGlobals.h File Reference

Include global constants and typedefs.

Functions

- typedef NS_ENUM (NSInteger, LogType)

  This class enumerates the MigratoryData logging levels.

Variables

- NSString * NOTIFY_SERVER_UP

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

  Indicate that the client failed to connect to a MigratoryData server.
- NSString * 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.
- NSString * 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.
- NSString * NOTIFY_SUBSCRIBE_ALLOW

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

  Indicate that the client was not authorized to subscribe to a subject.
- NSString * NOTIFY_PUBLISH_DENIED

  Indicate that the client was unable to publish a message because it is not allowed by your entitlement rules.
- NSString * **NOTIFY_SUBSCRIBE_TIMEOUT**
- NSString * 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.
- NSString * NOTIFY_PUBLISH_OK

  Indicate that the client successfully published a message.
- NSString * NOTIFY_PUBLISH_FAILED

  Indicate that the client was unable to publish a message.
- NSString * CONSTANT_WINDOW_BACKOFF

  A constant used to define the reconnect policy.
- NSString * TRUNCATED_EXPONENTIAL_BACKOFF

  A constant used to define the reconnect policy.

7.3.1 Detailed Description

Include global constants and typedefs.

7.3.2 Function Documentation

7.3.2.1 NS_ENUM()

```
typedef NS_ENUM (
            NSInteger ,
            LogType  )
```

This class enumerates the MigratoryData logging levels.

The available logging levels ordered by verbosity are:

- LOG_ERROR (less verbose)
- LOG_INFO
- LOG_DEBUG
- LOG_TRACE (most verbose)

For production usage, we recommend the default `LOG_INFO` logging level. The `LOG_ERROR` level turns on the error logs of the API.

The `LOG_INFO` level turns on the info, warning, and error logs of the API.

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

The `LOG_TRACE` level turns on all the logs of the API.

7.3.3 Variable Documentation

7.3.3.1 NOTIFY_SERVER_UP

```
NSString* 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:.

7.3.3.2 NOTIFY_SERVER_DOWN

```
NSString* 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:.

7.3.3.3 NOTIFY_DATA_SYNC

```
NSString* 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.

7.3.3.4 NOTIFY_DATA_RESYNC

```
NSString* 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.

7.3.3.5 NOTIFY_SUBSCRIBE_ALLOW

```
NSString* 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.

7.3.3.6 NOTIFY_SUBSCRIBE_DENY

```
NSString* 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 MigratoryDataClient↩::setEntitlementToken: – is not allowed to subscribe to the subject provided in the detail information of the status notification.

7.3.3.7 NOTIFY_PUBLISH_DENIED

```
NSString* NOTIFY_PUBLISH_DENIED
```

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

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.

7.3.3.8 NOTIFY_PUBLISH_NO_SUBSCRIBER

```
NSString* 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.

Deprecated no more is use.

7.3.3.9 NOTIFY_PUBLISH_OK

```
NSString* 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.

7.3.3.10 NOTIFY_PUBLISH_FAILED

```
NSString* 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.

7.3.3.11 CONSTANT_WINDOW_BACKOFF

```
NSString* CONSTANT_WINDOW_BACKOFF
```

A constant used to define the reconnect policy.

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

7.3.3.12 TRUNCATED_EXPONENTIAL_BACKOFF

```
NSString* TRUNCATED_EXPONENTIAL_BACKOFF
```

A constant used to define the reconnect policy.

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

7.4 src/MigratoryDataListener.h File Reference

Include the declaration of the MigratoryDataListener class.

Classes

- protocol <MigratoryDataListener>

  Implementations of this interface can handle the real-time messages received for the subscribed subjects as well as various status notifications.

7.4.1 Detailed Description

Include the declaration of the MigratoryDataListener class.

7.5 src/MigratoryDataMessage.h File Reference

Include the declaration of the MigratoryDataMessage class.

Classes

- class MigratoryDataMessage

  Represent a message.

7.5.1 Detailed Description

Include the declaration of the MigratoryDataMessage class.

Index