MigratoryData Client API for Java

Developer's Guide and Reference Manual

June 5, 2022

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 Java clients for MigratoryData Server

A typical API usage is as follows:

1.2.1 Step 1 - Include the library

Import in your application the classes of this API as follows:

```
import com.migratorydata.client.*;
```

Also, include in the `class-path` of your application the API library `migratorydata-client-java.jar` located in the folder `lib` of this API package.

1.2.2 Step 2 - Define the listener for processing 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 the MigratoryData servers where to connect to

Use the API method MigratoryDataClient.setServers() to specify a list of one or more MigratoryData servers to which the client will connect to. 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 load balancing and failover. Supposing the MigratoryData server – to which the client connected – goes down, then the API will automatically reconnect 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 Migratory↵ DataClient.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 MigratoryDataClient.setServersDownBeforeNotify (int n)
    use notifyAfterReconnectRetries

Member MigratoryDataListener.NOTIFY_PUBLISH_NO_SUBSCRIBER
    no more in 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

Class Documentation

5.1 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

- MigratoryDataClient ()

    Create a MigratoryDataClient object.
- void setLogging (MigratoryDataLogLevel logLevel, File logFile, int logRotateLimit) throws IOException

    Configure the logging parameters.
- void setListener (MigratoryDataListener listener)

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

    Get the MigratoryDataListener object defined for handling real-time messages and status notifications.
- void setServers (String[ ] servers) throws UnknownHostException

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

    Connect to a MigratoryData cluster.
- void subscribe (List< String > subjects)

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

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

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

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

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

    Assign an authorization token to the client.
- Collection< String > getSubjects ()

    Return the list of subscribed subjects.
- void setServersDownBeforeNotify (int n)

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

- 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 disconnect ()

  Disconnect from the connected MigratoryData server and dispose the resources used by the connection.
- void publish (MigratoryDataMessage message) throws Exception

  Publish a message.
- void setQuickReconnectMaxRetries (int retries)

  Define the maximum number of retries for the Quick Reconnect failover 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 MigratoryDataListener.TRUNCATED_EXPONENTIAL_BACKOFF policy.
- void setInteractiveEventFilters (Set< InteractiveEventType > interactiveEventTypes, List< String > subject←Prefixes)

  Define a filter for receiving entitlement and subscription notifications for a group of subjects.
- void setExternalToken (String externalToken)

  Assign an external token to a client.
- void setTransport (String type)

5.1.1 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.

5.1.2 Member Function Documentation

5.1.2.1 setLogging()

```
void MigratoryDataClient.setLogging (
            MigratoryDataLogLevel logLevel,
            File logFile,
            int logRotateLimit ) throws IOException
```

Configure the logging parameters.

It is advisable to configure this first if you want to log as much as possible. The default log level is MigratoryData←LogLevel.INFO.

Parameters

logLevel	The particular MigratoryDataLogLevel configured as the logging threshold
logFile	The file used to output the logs. For Android applications, set this parameter to `null`.
logRotateLimit	Define the maximum file size in bytes of the logging file to be used before creating a new logging file (*log rotation*). To disable log rotation, set this parameter on `0`.

Exceptions

IOException	If there is an IO error while trying to configure the logging file.

5.1.2.2 setListener()

```
void MigratoryDataClient.setListener (
            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

5.1.2.3 getListener()

```
MigratoryDataListener MigratoryDataClient.getListener ( )
```

Get the MigratoryDataListener object defined for handling real-time messages and status notifications.

Returns

The instance of a class which implements the MigratoryDataListener interface defined with MigratoryData↩ Client.setListener()

5.1.2.4 setServers()

```
void MigratoryDataClient.setServers (
            String [] servers ) throws UnknownHostException
```

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:

```
client.setServers(new String[] {"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 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:
>
> `client.setServers(new String[] {"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 new selected server, a status notification MigratoryDataListener.NOTIFY_SERVER_DOWN will be triggered (unless you modify the number of failed attempts with MigratoryDataClient.setServersDownBefore↩ Notify()), 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 MigratoryDataListener.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 MigratoryDataListener.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 MigratoryDataListener.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`.

Exceptions

UnknownHostException	If the address of a MigratoryData server could not be determined

5.1.2.5 connect()

```
void MigratoryDataClient.connect ( )
```

Connect to a MigratoryData cluster.

This API call can be used to connect to one of the MigratoryData servers specified with MigratoryDataClient.set←Servers().

Please note that a connection is automatically made during the first subscription using the API call MigratoryData←Client.subscribe() or during the first publication using the API call MigratoryDataClient.publish().

Therefore, if the creation a MigratoryDataClient object is immediately followed by a subscribe or publish operation, then the use of this API call is not necessary. Otherwise, use this API call to connect to a MigratoryData cluster.

5.1.2.6 subscribe()

```
void MigratoryDataClient.subscribe (
            List< 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:

```
List<String> subjects = new ArrayList<String>();
subjects.add("/stocks/NYSE/IBM");
subjects.add("/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.

5.1.2.7 subscribeWithConflation()

```
void MigratoryDataClient.subscribeWithConflation (
            List< 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:

```
List<String> subjects = new ArrayList<String>();
subjects.add("/stocks/NYSE/IBM");
subjects.add("/stocks/Nasdaq/MSFT");
client.subscribeWithConflation(subjects, 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.

5.1.2.8 subscribeWithHistory()

```
void MigratoryDataClient.subscribeWithHistory (
            List< 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 `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 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 `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.

```
    client.subscribeWithHistory(Arrays.asList("/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.

5.1.2.9 unsubscribe()

```
void MigratoryDataClient.unsubscribe (
            List< 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.

5.1.2.10 setEncryption()

```
void MigratoryDataClient.setEncryption (
            boolean b )
```

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

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

5.1.2.11 setEntitlementToken()

```
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.

5.1.2.12 getSubjects()

```
Collection<String> MigratoryDataClient.getSubjects ( )
```

Return the list of subscribed subjects.

Returns

The list of strings representing the subscribed subjects.

5.1.2.13 setServersDownBeforeNotify()

```
void MigratoryDataClient.setServersDownBeforeNotify (
            int n )
```

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

Deprecated use notifyAfterReconnectRetries

Parameters

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

5.1.2.14 notifyAfterReconnectRetries()

```
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`.

5.1.2.15 disconnect()

```
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.

5.1.2.16 publish()

```
void MigratoryDataClient.publish (
            MigratoryDataMessage message ) throws Exception
```

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

5.1.2.17 setQuickReconnectMaxRetries()

```
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`.

5.1.2.18 setQuickReconnectInitialDelay()

```
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, quickReconnectInitialDela

   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, reconnectTimeInte
```

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.

5.1.2.19 setReconnectPolicy()

```
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 MigratoryDataListener.CONSTANT_WINDOW_BACKOFF and MigratoryDataListener.TRUNCATED_EXPONENTIAL_BACKOFF; the default value is MigratoryDataListener.TRUNCATED_EXPONENTIAL_BACKOFF.

5.1.2.20 setReconnectTimeInterval()

```
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.

5.1.2.21 setReconnectMaxDelay()

```
void MigratoryDataClient.setReconnectMaxDelay (
             int seconds )
```

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

5.1.2.22 setInteractiveEventFilters()

```
void MigratoryDataClient.setInteractiveEventFilters (
            Set< InteractiveEventType > interactiveEventTypes,
            List< String > subjectPrefixes )
```

Define a filter for receiving entitlement and subscription notifications for a group of subjects.

The Coupling extension of the MigratoryData server is required in order to be able to use this function.

Parameters

interactiveEventTypes	Define the list of event types for which you will receive notifications. Currently, there are two types: entitlement notifications and subscription notifications.
subjectPrefixes	One or more strings representing the prefixes (the first subject segment) of the subjects for which you desire to receive notifications. For example, if you define "stocks" in this list, you will receive entitlement and subscriptions notifications for all subjects starting with the prefix "stocks", for example "/stocks/IBM", "/stocks/MFTS", etc.

5.1.2.23 setExternalToken()

```
void MigratoryDataClient.setExternalToken (
            String 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

5.1.2.24 setTransport()

```
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: MigratoryDataListener.TRANSPORT_HTTP and MigratoryDataListener.TRANSPORT_WEBSOCKET; the default value is the first one

5.2 MigratoryDataEntitlementEvent Interface Reference

Represent an entitlement event.

Public Member Functions

- String getToken ()

  Get the token.
- String getSubject ()

  Get the subject.
- void response (boolean response)

  Provide the entitlement response.
- void response (boolean response, String motive)

  Provide the entitlement response.

5.2.1 Detailed Description

Represent an entitlement event.

The Coupling extension of the MigratoryData server is required in order to be able to use this interface.

5.2.2 Member Function Documentation

5.2.2.1 getToken()

```
String MigratoryDataEntitlementEvent.getToken ( )
```

Get the token.

Returns

A string representing the token to authenticate a user.

5.2.2.2 getSubject()

```
String MigratoryDataEntitlementEvent.getSubject ( )
```

Get the subject.

Returns

A string representing the subject to which a user attempt subscribe or publish.

5.2.2.3 response() [1/2]

```
void MigratoryDataEntitlementEvent.response (
             boolean response )
```

Provide the entitlement response.

Parameters

response	A boolean representing the entitlement response. Use `true` to allow the user identified with the token provided by MigratoryDataEntitlementEvent.getToken() to subscribe or publish to the subject provided by MigratoryDataEntitlementEvent.getSubject().

5.2.2.4 response() [2/2]

```
void MigratoryDataEntitlementEvent.response (
             boolean response,
             String motive )
```

Provide the entitlement response.

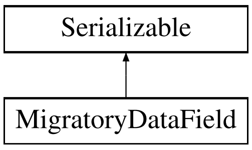
Parameters

response	A boolean representing the entitlement response. Use `true` to allow the user identified with the token provided by MigratoryDataEntitlementEvent.getToken() to subscribe or publish to the subject provided by MigratoryDataEntitlementEvent.getSubject().
motive	The motive for the entitlement refusal

5.3 MigratoryDataField Class Reference

Represent a message field.

Inheritance diagram for MigratoryDataField:



Public Member Functions

- MigratoryDataField (String name, String value)

  Create a MigratoryDataField object.
- String getName ()

  Get the field name.
- String getValue ()

  Get the field value.
- String toString ()

  Return a string representation of the message field.

5.3.1 Detailed Description

Represent a message field.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 MigratoryDataField()

```
MigratoryDataField.MigratoryDataField (
             String name,
             String value )
```

Create a MigratoryDataField object.

Parameters

name	The field name
value	The field value

5.3.3 Member Function Documentation

5.3.3.1 getName()

```
String MigratoryDataField.getName ( )
```

Get the field name.

Returns

A string representing the field name.

5.3.3.2 getValue()

```
String MigratoryDataField.getValue ( )
```

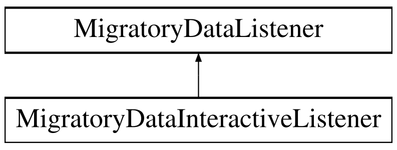
Get the field value.

Returns

A string representing the field value.

5.4 MigratoryDataInteractiveListener Interface Reference

Inheritance diagram for MigratoryDataInteractiveListener:



Classes

- enum **InteractiveEventType**

Public Member Functions

- void onSubscribe (String subject)

  Indicate a subscription to a subject by a user.
- void onUnsubscribe (String subject)

  Indicate an unsubscription to a subject by a user.
- void onEntitlementSubscribeCheck (MigratoryDataEntitlementEvent message)

  Indicate an entitlement check for subscription.
- void onEntitlementPublishCheck (MigratoryDataEntitlementEvent message)

  Indicate an entitlement check for publication.

Additional Inherited Members

5.4.1 Detailed Description

Implementations of this interface, like those of MigratoryDataListener, handle data messages and status notifications, as well as interactive events such as such as subscription and entitlement notifications.

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

The Coupling extension of the MigratoryData server is required in order to be able to use this interface.

Also, in order to get notifications along the methods of this interface, you must define one or more filters with MigratoryDataClient.setInteractiveEventFilters().

5.4.2 Member Function Documentation

5.4.2.1 onSubscribe()

```
void MigratoryDataInteractiveListener.onSubscribe (
             String subject )
```

Indicate a subscription to a subject by a user.

If multiple users subscribe to the same subject, only the fist subscription is notified.

Parameters

subject	The subject subscribed.

5.4.2.2 onUnsubscribe()

```
void MigratoryDataInteractiveListener.onUnsubscribe (
             String subject )
```

Indicate an unsubscription to a subject by a user.

If multiple users unsubscribe to the same subject, only the one last unsubscription is notified.

Parameters

subject	The subject subscribed.

5.4.2.3 onEntitlementSubscribeCheck()

```
void MigratoryDataInteractiveListener.onEntitlementSubscribeCheck (
             MigratoryDataEntitlementEvent message )
```

Indicate an entitlement check for subscription.

Parameters

message	An entitlement message

5.4.2.4 onEntitlementPublishCheck()

```
void MigratoryDataInteractiveListener.onEntitlementPublishCheck (
            MigratoryDataEntitlementEvent message )
```

Indicate an entitlement check for publication.

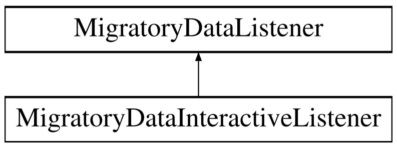
Parameters

message	An entitlement message

5.5 MigratoryDataListener Interface Reference

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

Inheritance diagram for MigratoryDataListener:



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 status, String info)

  This method handles the status notifications.

Static Public Attributes

- static final String NOTIFY_SERVER_UP = "NOTIFY_SERVER_UP"

  Indicate that the client successfully connected to a MigratoryData server.
- static final String NOTIFY_SERVER_DOWN = "NOTIFY_SERVER_DOWN"

  Indicate that the client failed to connect to a MigratoryData server.
- static final String NOTIFY_DATA_SYNC = "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.
- static final String NOTIFY_DATA_RESYNC = "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.
- static final String NOTIFY_SUBSCRIBE_ALLOW = "NOTIFY_SUBSCRIBE_ALLOW"

  Indicate that the client was authorized to subscribe to a subject.
- static final String NOTIFY_SUBSCRIBE_DENY = "NOTIFY_SUBSCRIBE_DENY"

  Indicate that the client was not authorized to subscribe to a subject.
- static final String **NOTIFY_SUBSCRIBE_TIMEOUT** = "NOTIFY_SUBSCRIBE_TIMEOUT"

- static final String NOTIFY_PUBLISH_OK = "NOTIFY_PUBLISH_OK"

  Indicate that the client successfully published a message.
- static final String NOTIFY_PUBLISH_FAILED = "NOTIFY_PUBLISH_FAILED"

  Indicate that the client was unable to publish a message.
- static final String NOTIFY_PUBLISH_DENIED = "NOTIFY_PUBLISH_DENIED"

  Indicate that the client was unable to publish a message because it is not allowed by your entitlement rules.
- static final String NOTIFY_PUBLISH_NO_SUBSCRIBER = "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.
- static final String SERVICE_DESTROYED = "SERVICE_DESTROYED"

  A constant holding the status type which indicates that the Android service running the push notifications service has been destroyed.
- static final String SERVICE_STOPPED = "SERVICE_STOPPED"

  A constant holding the status type which indicates that the push notifications service has been stopped.
- static final String SERVICE_DOWN_NO_NETWORK = "SERVICE_DOWN_NO_NETWORK"

  A constant holding the status type which indicates that the push notifications service is down.
- static final String SERVICE_START = "SERVICE_START"

  A constant holding the status type which indicates that the push notifications service is up.
- static final String SERVICE_RUNNING = "SERVICE_RUNNING"

  A constant holding the status type which indicates that the push notifications service is running.
- static final String INFO_ERROR_INVALID_SERVER = "invalid server for client configuration"

  A constant holding the info type which indicates that the push notification service has stopped because the list of servers provided with the method MigratoryDataClient.setServers() is null, empty or invalid. The info type is used with status notification MigratoryDataListener.SERVICE_STOPPED .
- static final String INFO_ERROR_INVALID_SUBJECT = "invalid subject for client configuration"

  A constant holding the info type which indicates that the push notification service has stopped because the list of subjects is null, empty or invalid. The info type is used with status notification MigratoryDataListener.SERVICE_S↩ TOPPED .
- 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.

5.5.1 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.

5.5.2 Member Function Documentation

5.5.2.1 onMessage()

```
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 .

5.5.2.2 onStatus()

```
void MigratoryDataListener.onStatus (
            String status,
            String info )
```

This method handles the status notifications.

The possible values of the `status` parameter are:

- `MigratoryDataListener.NOTIFY_SERVER_UP` indicates that the client successfully connected to the MigratoryData server provided in the detail information of the status notification
- `MigratoryDataListener.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
- `MigratoryDataListener.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.
- `MigratoryDataListener.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.
- `MigratoryDataListener.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
- `MigratoryDataListener.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
- `MigratoryDataListener.NOTIFY_PUBLISH_OK` indicates that the client successfully published the message having the closure data provided in the detail information of the status notification
- `MigratoryDataListener.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
- `MigratoryDataListener.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.set↩ EntitlementToken()` – is not allowed to publish on the subject of the message
- `MigratoryDataListener.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.

5.5.3 Member Data Documentation

5.5.3.1 NOTIFY_SERVER_UP

```
final String MigratoryDataListener.NOTIFY_SERVER_UP = "NOTIFY_SERVER_UP"  [static]
```

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().

5.5.3.2 NOTIFY_SERVER_DOWN

```
final String MigratoryDataListener.NOTIFY_SERVER_DOWN = "NOTIFY_SERVER_DOWN"  [static]
```

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().

5.5.3.3 NOTIFY_DATA_SYNC

```
final String MigratoryDataListener.NOTIFY_DATA_SYNC = "NOTIFY_DATA_SYNC"  [static]
```

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.

5.5.3.4 NOTIFY_DATA_RESYNC

```
final String MigratoryDataListener.NOTIFY_DATA_RESYNC = "NOTIFY_DATA_RESYNC"  [static]
```

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.

5.5.3.5 NOTIFY_SUBSCRIBE_ALLOW

```
final String MigratoryDataListener.NOTIFY_SUBSCRIBE_ALLOW = "NOTIFY_SUBSCRIBE_ALLOW"  [static]
```

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.

5.5.3.6 NOTIFY_SUBSCRIBE_DENY

```
final String MigratoryDataListener.NOTIFY_SUBSCRIBE_DENY = "NOTIFY_SUBSCRIBE_DENY"  [static]
```

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.

5.5.3.7 NOTIFY_PUBLISH_OK

```
final String MigratoryDataListener.NOTIFY_PUBLISH_OK = "NOTIFY_PUBLISH_OK"  [static]
```

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.

5.5.3.8 NOTIFY_PUBLISH_FAILED

```
final String MigratoryDataListener.NOTIFY_PUBLISH_FAILED = "NOTIFY_PUBLISH_FAILED"  [static]
```

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.

5.5.3.9 NOTIFY_PUBLISH_DENIED

```
final String MigratoryDataListener.NOTIFY_PUBLISH_DENIED = "NOTIFY_PUBLISH_DENIED"  [static]
```

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.

5.5.3.10 NOTIFY_PUBLISH_NO_SUBSCRIBER

```
final String MigratoryDataListener.NOTIFY_PUBLISH_NO_SUBSCRIBER = "NOTIFY_PUBLISH_NO_SUBSCRI↩
BER"  [static]
```

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 in use

5.5.3.11 SERVICE_DESTROYED

```
final String MigratoryDataListener.SERVICE_DESTROYED = "SERVICE_DESTROYED"  [static]
```

A constant holding the status type which indicates that the Android service running the push notifications service has been destroyed.

This constant indicated that the Android service notification service has been killed by the operating system.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.12 SERVICE_STOPPED

```
final String MigratoryDataListener.SERVICE_STOPPED = "SERVICE_STOPPED"  [static]
```

A constant holding the status type which indicates that the push notifications service has been stopped.

This constant indicated that the push notifications service is stopped.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.13 SERVICE_DOWN_NO_NETWORK

```
final String MigratoryDataListener.SERVICE_DOWN_NO_NETWORK = "SERVICE_DOWN_NO_NETWORK"  [static]
```

A constant holding the status type which indicates that the push notifications service is down.

This constant indicated that the push notifications service has been stopped because the phone has no network connectivity.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.14 SERVICE_START

```
final String MigratoryDataListener.SERVICE_START = "SERVICE_START"  [static]
```

A constant holding the status type which indicates that the push notifications service is up.

This constant indicated that the push notifications service has been started.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.15 SERVICE_RUNNING

```
final String MigratoryDataListener.SERVICE_RUNNING = "SERVICE_RUNNING"  [static]
```

A constant holding the status type which indicates that the push notifications service is running.

This constant indicated that the push notifications service is running.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.16 INFO_ERROR_INVALID_SERVER

```
final String MigratoryDataListener.INFO_ERROR_INVALID_SERVER = "invalid server for client
configuration"  [static]
```

A constant holding the info type which indicates that the push notification service has stopped because the list of servers provided with the method MigratoryDataClient.setServers() is null, empty or invalid. The info type is used with status notification MigratoryDataListener.SERVICE_STOPPED .

This constant indicated that the list of servers is invalid.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.17 INFO_ERROR_INVALID_SUBJECT

```
final String MigratoryDataListener.INFO_ERROR_INVALID_SUBJECT = "invalid subject for client
configuration"  [static]
```

A constant holding the info type which indicates that the push notification service has stopped because the list of subjects is null, empty or invalid. The info type is used with status notification MigratoryDataListener.SERVICE_↩ STOPPED .

This constant indicated that the list of subjects is invalid.

Attention

Used only in MigratoryData PushNotification API for Android

5.5.3.18 CONSTANT_WINDOW_BACKOFF

```
final String MigratoryDataListener.CONSTANT_WINDOW_BACKOFF = "CONSTANT_WINDOW_BACKOFF"  [static]
```

A constant used to define the reconnect policy.

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

5.5.3.19 TRUNCATED_EXPONENTIAL_BACKOFF

```
final String MigratoryDataListener.TRUNCATED_EXPONENTIAL_BACKOFF = "TRUNCATED_EXPONENTIAL_BA↩
CKOFF"  [static]
```

A constant used to define the reconnect policy.

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

5.5.3.20 TRANSPORT_HTTP

```
String MigratoryDataListener.TRANSPORT_HTTP = "TRANSPORT_HTTP"  [static]
```

A constant used to define the transport type.

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

5.5.3.21 TRANSPORT_WEBSOCKET

```
String MigratoryDataListener.TRANSPORT_WEBSOCKET = "TRANSPORT_WEBSOCKET"  [static]
```

A constant used to define the transport type.

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

5.6 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.

5.6.1 Detailed Description

This class enumerates the MigratoryData logging levels.

The available logging levels ordered by verbosity are:

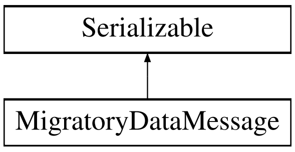
- ERROR (less verbose)
- WARN
- INFO
- DEBUG
- TRACE (most verbose)

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

5.7 MigratoryDataMessage Class Reference

Represent a message.

Inheritance diagram for MigratoryDataMessage:



Public Member Functions

- MigratoryDataMessage (String subject, String content)

  Create a MigratoryDataMessage object.
- MigratoryDataMessage (String subject, String content, String closure)

  Create a MigratoryDataMessage object.
- MigratoryDataMessage (String subject, String content, List< MigratoryDataField > fields)

  Create a MigratoryDataMessage object.
- MigratoryDataMessage (String subject, String content, List< MigratoryDataField > fields, String closure)

  Create a MigratoryDataMessage object.
- String getSubject ()

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

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

  Get the fields of the message.
- Map< 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)
- String getReplyToSubject ()
- String toString ()

  Return a string representation of the message.

Protected Attributes

- Map< String, String > **fieldMap**
- boolean **isSnapshot**
- boolean **isRecovery**
- int **seq**
- int **epoch**

5.7.1 Detailed Description

Represent a message.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 MigratoryDataMessage() [1/4]

```
MigratoryDataMessage.MigratoryDataMessage (
            String subject,
            String content )
```

Create a MigratoryDataMessage object.

Parameters

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

5.7.2.2 MigratoryDataMessage() [2/4]

```
MigratoryDataMessage.MigratoryDataMessage (
            String subject,
            String content,
            String closure )
```

Create a MigratoryDataMessage object.

Parameters

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

5.7.2.3 MigratoryDataMessage() [3/4]

```
MigratoryDataMessage.MigratoryDataMessage (
            String subject,
            String content,
            List< MigratoryDataField > fields )
```

Create a MigratoryDataMessage object.

Parameters

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

5.7.2.4 MigratoryDataMessage() [4/4]

```
MigratoryDataMessage.MigratoryDataMessage (
            String subject,
            String content,
            List< MigratoryDataField > fields,
            String 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

5.7.3 Member Function Documentation

5.7.3.1 getSubject()

```
String MigratoryDataMessage.getSubject ( )
```

Get the subject of the message.

Returns

A string representing the subject of the message

5.7.3.2 getContent()

```
String MigratoryDataMessage.getContent ( )
```

Get the content of the message.

Returns

A string representing the content of the message

5.7.3.3 getFields()

```
List<MigratoryDataField> MigratoryDataMessage.getFields ( )
```

Get the fields of the message.

Returns

The fields of the message as a list of MigratoryDataField objects

5.7.3.4 getFieldsAsMap()

```
Map<String, String> MigratoryDataMessage.getFieldsAsMap ( )
```

Get the fields of the message.

Returns

The fields of the message as a map.

5.7.3.5 getClosure()

```
String MigratoryDataMessage.getClosure ( )
```

Get the closure of the message.

Returns

The closure data of the message

5.7.3.6 isSnapshot()

```
boolean MigratoryDataMessage.isSnapshot ( )
```

Test whether the message is a snapshot message or not.

Returns

true if the message is a snapshot message

5.7.3.7 setReplyToSubject()

```
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.

5.7.3.8 getReplyToSubject()

```
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