MTS SDK Java Integration Guide

November 2018

betradar
driven by facts

Document version

Version	Author	Date	Comments
1.4	David Hrovat	2018-10-11	Added exclusiveConsumer configuration property
1.3	David Hrovat	2017-11-10	Added port property to settings
1.2	David Hrovat	2017-09-12	Added provideAdditionalMarketSpecifiers property to settings
1.1	Uros Bregar	2017-09-09	Added description about Rest log Added accessToken configuration property Updated method SetIdUof
1.0	Uros Bregar	2015-04-13	Initial version

Sportradar AG
www.sportradar.com

Feldlistrasse 2
CH-9000 St. Gallen
Switzerland

P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com

Company Identification
No:CHE-113.075.404

Table of Contents

Sportradar AG
www.sportradar.com

Feldlistrasse 2
CH-9000 St. Gallen
Switzerland

P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com

Company Identification
No:CHE-113.075.404

Getting started

Before starting to use the SDK please read the appropriate MTS documentation found on www.betradar.com under Help/Developer Zone/Downloads. SDK examples and code documentation is available on the https://sdk.sportradar.com.

Logging

To more easily find the log entry associated with a specific action that occurred within the SDK, the logs are split into several files.

- Traffic log: contains log entries for all messages send to or received from the MTS
- Rest log: contains log entries for all messages send or received from API calls[1]
- Execution log: contain log entries for all important actions and all error / warning conditions which occur within the SDK

For more information about the configuration of the SDK's logging module please refer to configuration section of this document.

Log files are used by the support team, so it is recommended to send them along with any issue related e-mails.

[1] REST API calls are made only when building ticket with UF selections

Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

Configuration

The SDK configuration can be specified in three different ways when opening the `MtsSdk` instance.

- mtsSdk.open() : Attempts to load the configuration from the default configuration file `mts-sdk.properties`.
- mtsSdk.open(filePath): Attempts to load the configuration from the configuration file specified by the file path.
- mtsSdk.open(properties); Attempts to load the configuration form the provided Properties instance

Note: all properties are written without quotation marks.

```
mts.sdk.username=username
mts.sdk.password=password
mts.sdk.hostname= mtsgate-ci.betradar.com
mts.sdk.vhost=/vhost
mts.sdk.ssl=true
mts.sdk.node=3
mts.sdk.bookmakerId=1
mts.sdk.limitId=1
mts.sdk.currency=EUR
mts.sdk.channel=INTERNET
mts.sdk.accessToken=your_uf_access_token
mts.sdk.provideAdditionalMarketSpecifiers=true
mts.sdk.port=5671
mts.sdk.exclusiveConsumer=true
```

Required attributes:

- `mts.sdk.username`: Username used to connect to the AMQP broker. Betradar provides this value.
- `mts.sdk.password`: Password used to connect to the AMQP broker. Betradar provides this value.
- `mts.sdk.hostname`: The hostname of the AMQP broker. Please use the following hostnames unless the integration team provides different ones.
    - Integration environment: mtsgate-ci.betradar.com

Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

- Production environment: mtsgate-t1.betradar.com

Optional attributes:

- `mts.sdk.vhost`: The name of the virtual host configured on the AMQP broker. If the value is not specified the value of '/username' attribute is used as virtual host.
- `mts.sdk.ssl`: The value specifying whether SSL will be used when connecting to the broker. Default value is `true`.
- `mts.sdk.node`: This value is used to filter MTS responses which were produced as responses to requests send by different SDK instances. In most configurations each SDK should use different node value. Default value is `1`.
- `mts.sdk.bookmakerId`: When provided, it is used as the default value for the `BookmakerId` on the ticket. The value can be overridden when building the ticket. Betradar provides this value.
- `mts.sdk.limitId`: When provided, it is used as the default value for the `LimitId` property on the ticket. The value can be overridden when building the ticket. Betradar provides the set of available values.
- `mts.sdk.currency`: When provided, it is used as the default value for the Currency property on the ticket. The value must comply with the ISO 4217 standard.
- `mts.sdk.channel`: When provided, it is used as the default value for the `SenderChannel` property on the ticket. Value must be one of the `SenderChannel` enumeration members.
- `mts.sdk.accessToken:` When selections are build using UnifiedOdds ids, the accessToken is used to access sports API. Also ensure that server running the sdk is whitelisted on api.betradar.com. Betradar provides this value.
- `mts.sdk.provideAdditionalMarketSpecifiers`: This value is used to indicate if the sdk should add market specifiers for specific markets. Only used when building selection using UnifiedOdds ids. If this is set to true and the user uses UOF markets, when there are special cases (market 215, or $score in SOV/SBV template), sdk automatically tries to add appropriate specifier; if set to false, user will need to add this manually.
- `mts.sdk.port`: Port should be chosen through the ssl property. Manually setting port number should be used only when non-default port is required.
- `mts.sdk.exclusiveConsumer`: The value specifying whether the rabbit consumer channel should be exclusive. Default value is `true`.

For more information about the ticket properties please refer to the MTS_Ticket_Integration document.

Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

Obtaining the SDK

The SDK is provided as a code library available for download on the SDK site. The archive contains three jar files using different packaging.

- mts-sdk-version-fatjar.shaded.jar – A shaded jar with dependencies
- mts-sdk-version-fatjar.jar – A non-shaded jar with dependencies
- mts-sdk-version-tinyjar.jar – A jar file without dependencies. Use the pom.xml file found in the archive to specify the SDK's dependencies

SDK setup and teardown

The SDK is setup by the following steps:

- Creating an instance of the `MtsSdk` class.
- Opening the created `MtsSdk` instance using one of the open(…) methods.
- Obtaining various sender objects, which can be used to send messages to the MTS

These steps can be performed by the following code:

```
MtsSdkApi mtsSdk = new MtsSdk();
mtsSdk.open();
BuilderFactory builderFactory = mtsSdk.getBuilderFactory();
TicketAckSender ticketAckSender = mtsSdk.getTicketAcknowledgmentSender(new
TicketAckHandler());
TicketCancelAckSender ticketCancelAckSender =
mtsSdk.getTicketCancelAcknowledgmentSender(new TicketCancelAckHandler());
TicketCancelSender ticketCancelSender = mtsSdk.getTicketCancelSender(new TicketCan-
celResponseHandler(ticketCancelAckSender, builderFactory));
TicketSender ticketSender = mtsSdk.getTicketSender(new Ticket-
ResponseHandler(ticketCancelSender, ticketAckSender, builderFactory));
```

For more information on how to implement listener callback methods please refer to the SDK examples and/or the SDK code documentation.

Once the initialized `MtsSdk` instance is no longer needed, it must be teardown in order to release resources held by it. This can be accomplished by the following method call:

```
mtsSdk.close();
```


Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

Building ticket instances

The SDK uses a "builder pattern" to simplify the process of creating new ticket instances. Below is the list of most noticeable builders.

- `TicketBuilder`: A root builder used as a starting point when building tickets.
- `SenderBuilder`: Used to specify the information about a ticket sender (bookmaker).
- `EndCustomerBuilder`: Used to build `EndCustomer` instances, representing the punter associated with the ticket. This information is part of the send element.
- `BetBuilder`: Used to build bet instances, which is part of the ticket. Each ticket must contain at least one bet.
- `SelectionBuilder`: Used to build selection instances, which are parts of bet. Each bet must contain at least one selection.

Below is a code snippet, which builds a ticket containing the mandatory information. Please note that some information from the configuration gets automatically applied to the ticket, so changing the configuration can make the snippet below produce an incomplete ticket. For more information refer to configuration section of this document and to MTS_Ticket_Integration document.

Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

```
Ticket ticket = builderFactory.createTicketBuilder()
        .setTicketId("T-" + System.currentTimeMillis())
        .setOddsChange(OddsChangeType.ANY)
        .setSender(builderFactory.createSenderBuilder()
                .setBookmakerId(Constants.BOOKMAKER_ID)
                .setLimitId(Constants.LIMIT_ID)
                .setSenderChannel(SenderChannel.INTERNET)
                .setCurrency("EUR")
                .setEndCustomer(builderFactory.createEndCustomerBuilder()
                        .setIp("127.0.0.1")
                        .setId("Customer1")
                        .setLanguageId("EN")
                        .setDeviceId("device1")
                        .setConfidence(12092)
                        .build())
                .build())
        .addBet(
                builderFactory.createBetBuilder()
                .setBetId("Bet-" + System.currentTimeMillis())
                .addSelectedSystem(1)
                .setStake(50000, StakeType.UNIT)
                .addSelection(
                        builderFactory.createSelectionBuilder()
                        .setEventId(9738581)
                        .setId("lcoo:43/1/*/YES")
                        .setOdds(14800)
                        .setBanker(false)
                        .build())
                .build()
        )
        .build();
```


Sportradar AG
www.sportradar.com

Feldlistrasse 2
CH-9000 St. Gallen
Switzerland

P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com

Company Identification
No:CHE-113.075.404

Sending tickets to MTS

SDK supports two ways of sending tickets to the MTS. The recommended way is to use the non-blocking mode. Non-blocking indicates the execution of the current thread is not blocked after the ticket is send and the response from MTS is processed in another thread. To send a ticket in a non-blocking mode, the following call can be used:

```
ticketSender.send(ticket);
```

When sending the ticket in the blocking mode, the current thread is blocked until a response from MTS is received or a timeout occurs (15 seconds). When using the blocking mode the responseReceived(…) callback method on the listener is not invoked. Ticket can be send in a blocking mode using the following statement:

```
TicketResponse ticketResponse = ticketSender.sendBlocking(ticket);
```


Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404

Tips and tricks

Building selection instances

The SDK supports markets used by three Betradar feeds – LO (Live Odds), LCoO (Live Cycle of Odds) and UF (Unified Feed) implemented by different methods on the `SelectionBuilder` type.

- `setId(String id);`
  This method should be used when building string representations of the market identifiers directly (without the help from the SDK).
- `setIdLo(int type, int subType, String sov, String selectionId);`
  This method should be used when building market identifiers from information provided by the LO feed.
- `setIdLcoo(int type, int sportId, String sov, String selectionId);` This method should be used when building market identifiers from information provided by the LCoO feed.
- `setIdUof(int product, String sportId, int marketId, String selectionIds, Map<String, String> specifiers, Map<String, Object> sportEventStatus);`
  This method should be used when building market identifiers from information provided by the UF feed. Note: this method will throw if accessToken is not provided.
  Method parameter sportEventStatus needs the following keys:
  - HomeScore (home_score in sport event status)
  - AwayScore (away_score in sport event status)
  - Server (current_server in sport event status)

  If you are using UnifiedFeed sdk the map with the correct keys may be obtained:
  Map<String, Object> sportEventStatus = competition.getStatus().toKeyValueStore();

Sportradar AG
www.sportradar.com
Feldlistrasse 2
CH-9000 St. Gallen
Switzerland
P: +41 71 517 72 00
F: +41 71 517 72 99
E: sales@sportradar.com
Company Identification
No:CHE-113.075.404