Johannesburg Stock Exchange

Post-trade Services

JSE Services Documentation
Volume PT01 – Post-trade EMAPI Common

Document Version	2.8.3
RTC Version	1.37.0
Release Date	17 March 2023
Number of Pages	77 (Including Cover Page)

1 DOCUMENT CONTROL

1.1 Table of Contents

1.2 Document Information

Drafted By	Post-trade Services
Status	FINAL
Version	2.8.3
Release Date	17 March 2023

1.3 Revision History

Date	Version	Description
11 May 2016	1.0	Initial draft created.
13 July 2016	1.1	Document updated to include changes for RTC Release 1.7.0 and 1.8.0. Following is a summary of the changes applied: 1. Added note to `MsgType` field of Header message (Section 4.1) 2. Updated description of Response types (Section 5.1) 3. Updated definition of String (Section 3.4.3) 4. Deleted Sections 6.4.3 (On Behalf Of) and 6.4.4 (Gateway User) as functionality has been removed from RTC. On-behalf-of functionalty will be delivered in a subsequent release. Clearing members will be given a system role to perform on behalf actions for their trading members. 5. Removed redundant field `reverseInfo` from tables where it was listed as field has been removed from the API (e.g. Section 7.3.2) 6. Updated Section 7.3.3 (Synchronise Subscription/Replay) – it incorrectly stated that replay subscription was not supported 7. Updated Section 7.4 (Subscription Termination) to indicate that SubscriptionTerminationEvent will not be sent to subscribers 8. Updated Appendix A to reflect message defintions as per latest RTC version
28 Sep 2016	1.2	Summary of updates for RTC Release 1.9.0 and 1.10.0. See EmapiTransactionsRevHistForMember for more details: 1. The SegmentSize of `TaxReplayReq` not used in RTC 2. Field description changes to the SubscriptionGroup, Member and AccessGroup objects.
25 Oct 2016	1.3	1. Added Section 7.6 on building a copy of the reference data cache. Used to identify when reference data is added, updated or deleted from RTC.
9 Dec 2016	1.4	1. Updated `Warning` description of response messages in section 5.1

Date	Version	Description
20 Jan 2017	1.5	Updated for RTC Release 1.13.0: 1. Updates to Section *7.6 Building a copy of reference data cache.*
17 Feb 2017	1.6	Added new *Section 5.3 Bookmarking or Pagination* to highlight existing bookmarking functionality more clearly to users. New message `TaxSessionStatus` added to provide additional information on session terminations.
10 Mar 2017	1.7	Minor cosmetic changes. No functional changes were introduced in this version of the document.
03 June 2017	1.8	Added a note to *Section 5.3 Bookmarking or Pagination* regarding the use of the bookmark field.
30 June 2017	1.9	Minor cosmetic changes. No functional changes were introduced in this version of the document.
21 July 2017	2.0	Updates for RTC Release 1.19.1: 1. Updated Section 9.1.1 to highlight how Recovery and Outstanding Request functionality is handled. Appendix B was added for a full list of messages and how they will respond to outstanding requests. 2. Updates to Section 5.2 *Message Responses.* 3. Field description changes to the TaxLogonReq message and Member object.
14 Oct 2017	2.1	Updates for RTC Release 1.20: 1. Field description changes for the TaxSnapshotSubscribeRsp message 2. Removed information regarding the lastPublishedSeqNo in Section 7.3.2 and 7.3.4.
22 Jan 2018	2.2	Updates for RTC Release 1.21: 1. Field description changes for the TaxEndSnapshot message.
05 Mar 2018	2.3	Minor cosmetic changes. No functional changes were introduced in this version of the document.
03 Apr 2018	2.4	Only version number updates
13 Jul 2018	2.5	Only version number updates
13 Aug 2018	2.6	1. Updated **`MARKET_DATA_FLOW`** Replay column in section 7.1
04 Sept 2018	2.7	Only version number updates
25 Sept 2018	2.7.1	Added section 4.1.1 Setting a unique ClientTxRef
15 Nov 2018	2.7.3	Only RTC version number update in Appendix A
22 Feb 2019	2.7.4	RTC version update in Appendix A to 1.28.1
11 Jun 2019	2.7.5	Only version number updates
30 Jul 2019	2.7.6	Added section 5.6.2.1 with new encrypted TLS versions 1.2 and 1.3 protocols
27 August 2019	2.7.7	Update to RTC version number 1.30
29 November 2019	2.7.8	Update to RTC version number 1.31.0

Date	Version	Description
12 February 2020	2.7.9	Update to RTC version number 1.32.0 Also updated section 6.4.1 *Change Password* section to reflect correct password length
24 August 2020	2.8.0	Update to RTC version number 1.34.0. This version includes version 1.33.0
01 February 2022	2.8.1	Update to RTC version number 1.35.0
08 April 2022	2.8.2	Update to RTC version number 1.36.1
17 March 2023	2.8.3	Update to RTC version number 1.37.0

1.4 About this Document

The purpose of this document and its related documents (See1.7) is to serve as guidance to the EMAPI protocol when implementing EMAPI client applications or backend systems to integrate with the JSE's real-time clearing (RTC) system.

This document provides guidance on the common or session/admin part of the EMAPI protocol such as:

- basic concepts, e.g. versioning and types
- message header syntax
- administration messages, e.g. logon, password change, session termination
- subscriptions to broadcast flows
- reconciliation between client and server
- recovery and failover

Note: The complete EMAPI syntax is described in the related documents *EmapiTransactionsForMember.xml, EmapiTransactionsForMember.html* and *EmapiTransactions.xsd (See 1.7).*

The application/business messages in the protocol are described in *Volume 02 – Post-trade EMAPI Clearing* (See 1.7).

All EMAPI messages to and from RTC are encoded using the TagWire encoding format (please refer to the *EMAPI TagWire* document for the specifications for the encoding).

1.5 Intended Audience

The information in this document is intended for software developers writing EMAPI interfaces to RTC.

1.6 Typographical Conventions

EMAPI messages or enumerations are shown in upper camel case using the `courier new` font and are hyperlinked to their detailed definitions in Appendix A. For example: `TaxLogonReq` or `TaxSnapshotSubscribeReq`.

EMAPI fields are shown using the `courier new` font in lower camel case. For example: `requestType` or `flow`.

Note: For ease of navigation of the document using the hyperlinks, please ensure you have the *Previous View (Alt + Left Arrow)* and *Next View (Alt + Right Arrow)* buttons enabled on the page navigation toolbar of Adobe Acrobat or the equivalent in other PDF viewers (if availalble).

1.7 Related Documents

Note: The documents in the table below are published on the ITaC website:
https://www.jse.co.za/services/itac#PostTradeDocumentation

Name	Description
Volume PT02 – Post-trade EMAPI Clearing.pdf	Describes the semantics and syntax of the clearing or application messages of the EMAPI protocol.
EMAPI TagWire.pdf	Describes the syntax of the TagWire encoding of EMAPI messages body.
EmapTransactionsForMember.xml	XML definition of all EMAPI protocol messages for market participants, i.e. clearing and trading members.
EmapiTransactionsForMember.html	HTML file describing the syntax of all EMAPI protocol messages for market participants i.e. clearing and trading members.
EmapiTransactionsForMemberRevHist	New HTMLfile describing the revision history of changes to EMAPI specifications between RTC releases.
EmapiTransactions.xsd	The XML Schema that EmapiTransactionsForMember.xml must conform to.

1.8 Contact Details

JSE Limited
One Exchange Square
Gwen Lane, Sandown
South Africa
Tel: +27 11 520 7000

www.jse.co.za

Post Trade and Information Services

ITAC Queries
Email: CustomerSupport@jse.co.za

1.9 Definitions, Acronyms and Abbreviations

EMAPI	External Messaging API. EMAPI is the API used to integrate a client application or backend system with the RTC Clearing System.
cCran	An administration front-end for the RTC clearing system.
cDew	A monitoring front-end for the RTC clearing system.
Client	A client that connects to RTC Servers using the EMAPI protocol.
RTC	Real-time Clearing. The JSE implementation of Cinnober TradeExpress™ clearing system.
Server	RTC server that supports the EMAPI protocol. For example, the TAX (Trading Application Multiplexer) Server.

2 OVERVIEW

This document describes the semantics of the EMAPI protocol and the syntax of the common or session/admin EMAPI messages. The common part of the protocol is shared by all functional or business parts of the protocol used in the JSE post-trade Real-time Clearing System (RTC).

Note: *Volume PT02 – Post-trade EMAPI Clearing* describes the semantics and syntax of the clearing or application messages of the EMAPI protocol

Note: The complete EMAPI syntax is described in the related documents *EmapiTransactionsForMember.xml, EmapiTransactionsForMember.html* and *EmapiTransactions.xsd* (See1.7).

2.1 EMAPI Overview

EMAPI, short for **E**xternal **M**essaging **API**, is an API used to integrate a client application or backend system with RTC. EMAPI is a relatively low-level API whereby carrying out a business function may require several API calls. EMAPI also contains functionality for setting up subscriptions to system events that a client needs to process in order to get the full picture of the current market status.

2.2 Data Communication

EMAPI is a TCP/IP based API where the client connects to the RTC Server and authenticates with a supplied user ID and password. Once the session is established and authenticated, the client may use the available API functions to perform business tasks, subscribe to events, etc.

The API is asynchronous so the client application must map responses to outstanding requests to be able to determine the outcome of a certain command.

2.3 RTC System Architecture

The diagram below depicts a high-level RTC architecture, with its main sub-components and how it interfaces with other systems.

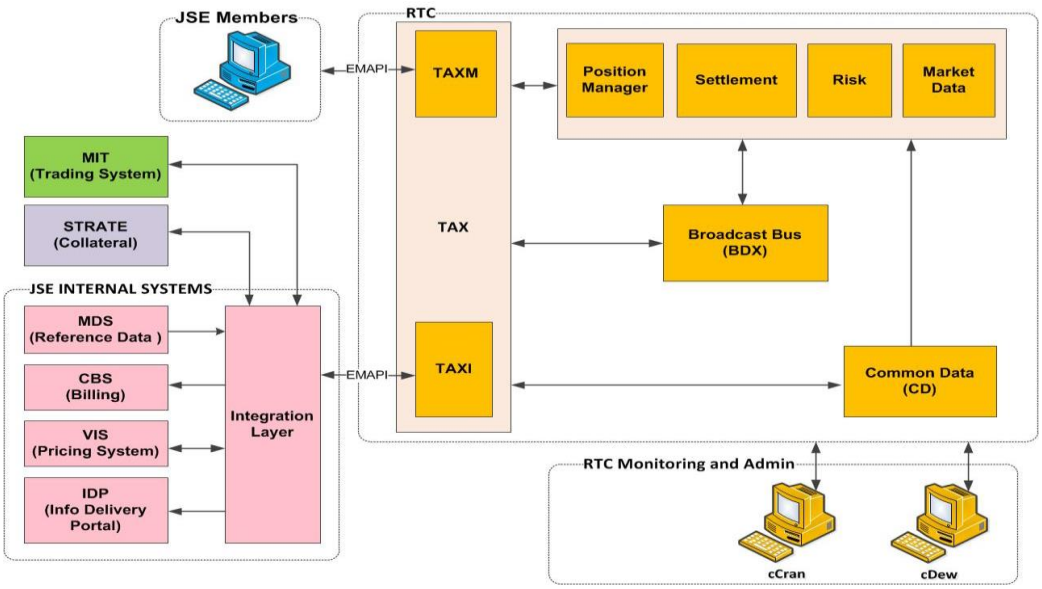

Figure 1 - RTC Architecture Overview

The RTC system provides two EMAPI interfaces:

- TAXM is used by JSE members to interface to RTC. This document deals with the API provided by this interface.
- TAXI is used internally by the JSE to connect the Trading System and other JSE Internal Systems to RTC. This interface is not available to JSE members.

RTC provides the following two front-ends that will be used exclusively by the JSE:

- cCran to administer the system, and
- cDew to monitor system status.

The RTC sub-components TAXM, TAXI, CD, BDX, Positions, Settlement, Risk, and Market Data are further described below.

2.3.1 Trade Application MultipleXer (TAXM, TAXI)

TAX stands for Trade Application MultipleXer. TAX is the first point of contact in RTC.

The JSE members will connect to the TAXM component. This interface has the following key functions:

- Receive messages using EMAPI protocol.
- Apply validation and embellishment on the messages.
- Reject erroneous messages with adequate responses.
- Route successful messages to the relevant internal RTC components depending on the message type.

2.3.2 Common Data (CD)

Common Data (CD) is a reference data component. Reference data within the CD component will be updated by JSE internal systems and published to other RTC sub-components. CD does not store historical reference data. Clients will be able to subscribe to some of this reference data via the TAXM interface.

2.3.3 Broadcast Bus (BDX)

Broadcast Bus (BDX) is used for publishing of data within RTC between the various internal components. BDX is not accessible directly by external components, but message flows in BDX are made accessible through the TAX.

2.3.4 Position Manager Component

The Position Manager (PM) comprises of trade (i.e. deal management). The PM component can receive requests from three key sources, namely Members via an external front end, JSE via the cCran front end, or JSE Integration Layer.

The PM component can be partitioned based on instrument groupings. This means that the request received from all three key sources will be routed to the same PM components for the various instrument groups.

2.3.5 Settlement Component

The settlement components gets trades and position updates from the Position Manager. The component then uses these messages to extracts all the payments and updates the members accounts based on the deals per member or per client. All settlement messages will be directed to Strate via the JSE integration layer. RTC will send these settlement instructions to JSE integration via EMAPI. JSE integration will generate the SWIFT message and send these instructions to Strate.

2.3.6 Risk Component

The Risk component performs risk calculations in real-time. Risk is calculated after the positions are updated. The risk component receives trade and position updates from the Position Manager component.

2.3.7 Market Data Component

The Market Data component receives external market data. The price engine is a component within the Market Data component that calculates various market related prices such as Mark-to-Model (MTM) prices. The Market Data Gateway within the Market Data component is a gateway within the RTC system for internal communications.

Note: Market data is published on the Market Data Event Flow. Please see *Section 7 Subscriptions* for details on how to subscribe to market data.

3 EMAPI VERSIONING

EMAPI has the following version numbers:

- `majorVersion`: An increase of this number implies that the EMAPI protocol has been changed in a fundamental way. For example, the client/server interaction model has changed.
- `minorVersion`: An increase of this number implies that the EMAPI protocol has been changed in a non-compatible way. For example, mandatory messages/fields have been added and/or messages/fields have been removed.
- `microVersion`: An increase of this number implies that the EMAPI protocol has been changed in a compatible way. For example, non-mandatory messages/fields have been added.

Note: The EMAPI version numbers used in the RTC installation is specified in the related documents EmapiTransactionsForMember.xml and EmapiTransactionsForMember.html (See 1.7)

The server verifies that the client's version is compatible with the version that the server has implemented. If the client's implemented version is not compatible with the server's implemented version, the session establishment (see 6.2) will fail.

A single version of EMAPI will be live in RTC at all times; therefore, any changes to the EMAPI version, irrespective of whether the changes are micro, minor or major, require clients to conform to that particular version of EMAPI.

Note: The JSE will communicate to clients the conformance requirements for each version.

4 MESSAGE HEADER

4.1 Message header

Each EMAPI message starts with a header as detailed below:

Name	Position	Length	Description
`magicSign`	0	4	Will be set to “XMMA” (0x58 0x4D 0x4D 0x41).
`headerVersion`	4	2	Will be set to “10” (0x31 0x00).
`msgSize`	6	6	Message size in bytes excluding this header. ASCII encoded. Example: “000100” means that the length of the message excluding this header is 100 bytes.
`clientTxRef`	12	4	Client assigned request ID. Binary network byte order. A response sent back by server will contain the same value as set in the request. In case the request is a subscription request, all events resulting from this subscription (as well as the subscription response) will contain the same value as set in the subscription request.
`msgType`	16	1	Message type: • `'R' (0x52)`: request sent by the client/response sent by the server. • `'B' (0x42)`: event sent by the server that is not part of a snapshot or replay. • `'S' (0x53)`: event sent by the server that is part of a snapshot. • `'H' (0x48)`: event sent by the server as part of a replay. **Note:** `'M' (0x4D)`: multicast event retransmitted on an EMAPI session. Not used in the RTC configuration for JSE.
`contentType`	17	1	Encoding used for the message body following this message header: • ‘W’ (0x57) = TagWire **Note:** The TagWire encodings are described in the *EMAPI TagWire* document (see 1.7).
`compressed`	18	1	Compressed message body flag: • `'Y' (0x59)` = Compressed message body • `' ' (0x20)` = Uncompressed message body **Note:** RTC does not support compression. If a message is sent compressed it will be rejected by RTC.
	19	1	Reserved. Will be set to ‘ ’ (0x20)

Table 1 - EMAPI Header

4.1.1 Setting a unique ClientTxRef

Setting a unique ClientTxRef for each request allows each request to be linked to its corresponding response which has the following benefits:

a) The client can handle failed responses and respond appropriately e.g. Re-send requests that have failed, identify which step in a process failed e.g. adding a non-resident client.
b) The client can identify the requests for which a corresponding response was not returned.
c) When the server concurrently processes several requests from a user, the responses to the requests might not be received by the client in the same order as the client sent in the requests.
d) Certain requests have a generic response message which is applicable to different types of requests which makes linking of the request to the response near impossible.

5 MESSAGES

All interaction between a client and a server is done using messages. All the messages syntax is specified in the related documents *EmapiTransactionsForMember.html*, *EmapiTransactionsForMember.xml* and *EmapiTransactions.xsd* (See 1.7).

Note: For convenience, the message definitions from *EmapiTransactionsForMember.html* used in this document are available in this document via hyperlinks to Appendix A- Message Formats.

5.1 Message Types

The following message types are available through EMAPI:

Message Type	Description
`Request`	Messages sent by a client to a server requesting RTC to perform one or several tasks. Requests are named `<task>Req`. For example, `TaxLogonReq`.
`Response`	Messages sent by a server to a client delivering the response to a previous request. The normal response for a request is specified in message specifications but usually has the `<task>Rsp` format. For example, `TaxLogonRsp`. **Note:** Two general response messages are possible: `ResponseMessage` is used for fundamental errors; `SimpleRsp` is used as a general response where no specific information is needed in the response and for some general errors. **Note:** All requests to RTC should be able to handle these two general response messages. The status code should be used to validate execution; this is independent of the response type.
`Event`	Messages sent by a server to a client to inform about events that have taken place in RTC. A client only receives events that it has subscribed to. See Section 7 for information regarding what information all events contain.

Table 2 - EMAPI Message Types

5.2 Message Responses

All response messages contain the following information:

Field Name	Description
`code`	• `Ok`: Indicates that the server has completed the processing of the request successfully. • `TaxSessionThrottled`: Indicates the same as `Ok` with the addition that the request has been throttled by the server, and thus the processing of the request has been delayed. • `RtcMessageAlreadyProcessed` and several other similar codes: Indicates that the server did not process this request, since an identical request had already been processed earlier. See section 9.1.1 for a full list of these codes.

	• `Warning`: o Indicates that the request has been successfully performed but some alert was raised. The alert text will be available in the Message below. OR o Indicates that a request is used to perform several tasks and RTC has successfully performed some of the tasks but has failed to perform others. The tasks that have failed will be signalled in the subCode below. • `TaxGateletFailure`: Indicates that the request might have been lost due to a failover between RTC internal servers. The request must be handled in the same way as outstanding requests at session recovery (i.e. the client must resend the concerned request with the `possDup` (=possible duplicate) field set. See section 9.1 for more information. • All other values are indicating that the server has failed to process the request.
`subCode`	Used when the request is to perform several tasks and the above `code` value returned is `warning`. The position of the `subCode` corresponds to the position of the task in the request message. For example: **Note**: The JSE will not be using `subCode`.
`Message`	A text description explaining in more detail why the request failed or resulted in a warning. **Note**: If the code is Ok the string will be "Ok".

Note: Other responses such as `SimpleRsp` or `CdResponse` may contain other key fields such `reply` and `latestSSN` respectively. Please see message specifications for more details.

5.3 Bookmarking or Pagination

For some messages (e.g. `GetPaymentAdvicesReq`) the operation can be performed any number of times. The response is paged (bookmarked), i.e. there will be an indication as to whether there is more information to be retrieved from RTC.

If the message supports bookmarking, it will have a field named `bookmark`. The `bookmark` marks a specific item in a list of data on the server. The `bookmark` received in the response should be used in the next request to get the next page of information.

Note: For the messages that make use of the bookmark field, only send the bookmark field when you already received a bookmark value in the response message. Do not send the bookmark field at all (null or blank) if you didn't receive a bookmark value before,

5.4 Primitive Types

5.4.1 Integers

Name	Value range (inclusive)
byte	-128 to 127 (-2^7 to 2^7-1)
short	-32 768 to 32 767 (-2^{15} to $2^{15}-1$)
int	-2 147 483 648 to 2 147 483 647 (-2^{31} to $2^{31}-1$)
long	−9 223 372 036 854 775 808 to 9 223 372 036 854 775 807(-2^{63} to $2^{63}-1$)
BigInteger	Arbitrary precision integer (i.e. value range is not applicable).

The types Long and Integer can have the same values as long and int, but can also have a null value.

5.4.2 Fixed-point number

Specified as an integer (see section 5.4.1) together with a scaling factor, which is the multiplicative inverse of the specified value of the enumeration DIVISOR.

5.4.3 Boolean

“Boolean” is a logical type that can be either true or false or null.

“boolean” is a logical type that can be either true or false.

5.4.4 String

String is a sequence of Unicode characters.

The maximum length of the string is specified in the message definition. If no length is specified, the maximum length of the string is 255 characters.

Note: RTC supports by default only a subset of characters with ASCII code points between 0 and 255.The allowed ASCII numbers for characters are 32-128 and 160-255.Derived Scalar Types

5.4.5 Derived Scalar Types

Enumeration types are supported.

5.4.6 Binary Type

Binary type is a sequence of bytes.

5.5 Composite Type

5.5.1 Records

A record is a sequence of fields that are indexed by their name.

- The fields may be of different data types.
- A field is either required or non-required.

Note: A message body is a record.

5.5.2 Arrays

An array is a sequence of fields that are indexed by their position. Arrays are specified with double brackets in the EMAPI message definitions. For example, `long[]` means an array of long values.

5.6 Example of Request / Response

In this example, a request `GetPaymentAdviceReq` to obtain payment advices (See *Volume PT02 – Post-trade EMAPI Clearing* for more details) is sent to the TAX server; the server then responds with a `GetPaymentAdviceRsp` message.

5.6.1 Sequence Diagram

The following diagram illustrates the typical flow of request-response interaction between a client and the RTC TAX server using a message to get payment advices.

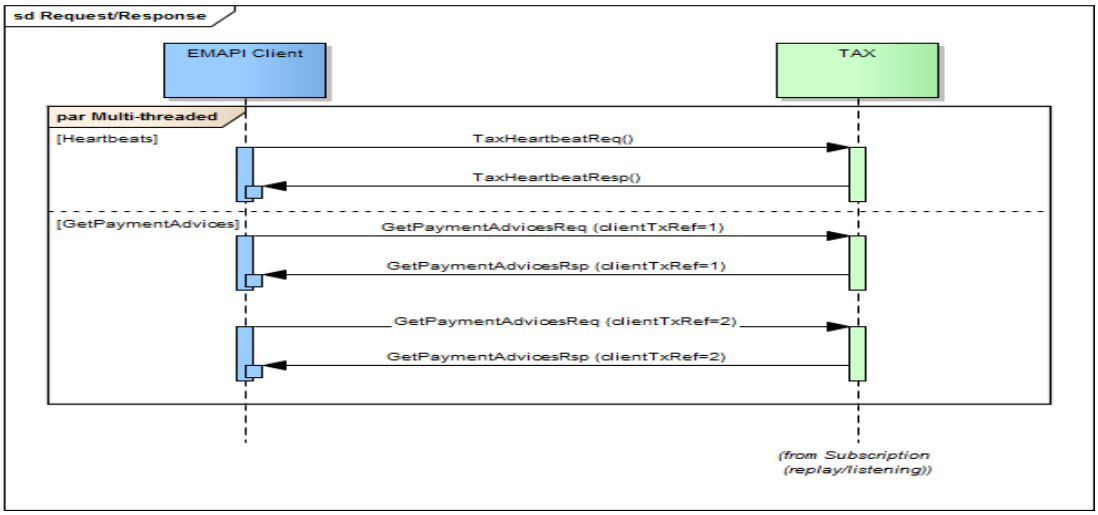

Figure 2 - Request/Response

Note: `possDup=true` signals RTC to check the message body in case it's a duplicate, i.e. it does not pertain to the header fields. Therefore, the `clientTxRef` field does not have to be the same if the message is being resent with `possDup=true`.

Note: `ResponseMessage`/`SimpleRsp` can be sent as a "successful" response for request messages that don't have a defined response message.

5.6.2 Connection to TAX Server

A single session has a single TCP socket; therefore, if the client has multiple requests running on the same session, responses may be returned out of order. Therefore, the client must either:

1. Submit requests on a single thread (one TCP socket), or
2. Handle out of order (one TCP socket), or
3. Have multiple sessions (multiple TCP sockets) that each have a single request thread.

In order to keep the TCP/IP connection alive, heartbeats must be sent to the TAX server at regular intervals. (See 6.3)

The communication with the TAX server is via asynchronous non-blocking calls; in addition, the server can run multiple threads in order to process concurrent requests.

The TAX server and the EMAPI protocol support low-latency and low-bandwidth utilisation resulting in near real-time performance.

5.6.3 5.6.2.1 Encrypted Secure TAX-M Connection

JSE will be provisioning an encrypted secure TAX-M connection channel, which will be available to all Client facing TAX-M environments in stages.

The secure TAX-M connection will take place over **TLS versions 1.2 and 1.3.**

To secure the transfer of data, TLS uses one or more cipher suites. A cipher suite is a combination of authentication, encryption, and message authentication code (MAC) algorithms. The Cipher suites will apply during negotiation of security settings for a TLS connection as well as for the transfer of data. As such, the below Table 1 Ciphers will be made available on the secure TAX-M connection.

Client's frontends must be updated to support the new TLS 1.2 and 1.3 protocol connection.

Note: JSE will be supporting both the unencrypted and the new encrypted channels for TAX-M connections for a certain period until further notice (as per the July 2019 Hotline).

Table1: TLS Ciphers

Order	Key Exchange Algorithm	Authentication Algorithm	Bulk Encryption Algorithm	Mac Algorithm
#1	Elliptic Curve Diffie–Hellman (ECDH)	Elliptic Curve Digital Signature Algorithm (ECDSA)	AES 256 in Galois Counter Mode (AES256-GCM)	SHA384
#2	Elliptic Curve Diffie–Hellman (ECDH)	RSA	AES 256 in Galois Counter Mode (AES256-GCM)	SHA384
#3	Elliptic curve Diffie–Hellman (ECDH)	Elliptic Curve Digital Signature Algorithm (ECDSA)	ChaCha20 (CHACHA20)	POLY1305
#4	Elliptic curve Diffie–Hellman (ECDH)	RSA	ChaCha20 (CHACHA20)	POLY1305
#5	Elliptic Curve Diffie–Hellman (ECDH)	Elliptic Curve Digital Signature Algorithm (ECDSA)	AES 128 in Galois Counter Mode (AES128-GCM)	SHA256
#6	Elliptic curve Diffie–Hellman (ECDH)	RSA	AES 128 in Galois Counter Mode (AES128-GCM)	SHA256
#7	Elliptic Curve Diffie–Hellman (ECDH)	Elliptic Curve Digital Signature Algorithm (ECDSA)	AES 256 (AES256)	SHA384
#8	Elliptic curve Diffie–Hellman (ECDH)	RSA	AES 256 (AES256)	SHA384
#9	Elliptic curve Diffie–Hellman (ECDH)	Elliptic Curve Digital Signature Algorithm (ECDSA)	AES 128 (AES128)	SHA256
#10	Elliptic curve Diffie–Hellman (ECDH)	RSA	AES 128 (AES128)	SHA256

5.6.4 Failure conditions and recovery

The client needs to be able to handle failure conditions and respond appropriately, e.g. by resending requests. Some failure conditions that are possible include:

- Malformed responses.
- Correlation failure (`clientTxRef` returned is unknown).
- Error response (expected Response message with error code or `ResponseMessage` or `SimpleRsp`).
- Timeouts.

6 ADMIN MESSAGE FLOWS

Interaction between an EMAPI client and a TAX server is done inside one or more user sessions with the following exceptions:

- when establishing a user session (see section 6.2), or
- changing the password (see section 6.4.1); this may be performed without having a session established.

If the servers are partitioned over several session types and/or subscription groups (see section 6.1), a user may need to establish several user sessions with different servers.

6.1 Server Session Types

The following server session types are provided by RTC:

- `INTEGRATION_TAX` (available to Internal JSE systems only)
- `MEMBER_TAX` (available to trading and clearing members)

6.2 Session Establishment and Authentication

A session is established by performing the steps described in this section.

6.2.1 Logon

The server that a client will use for a user session is determined from the information provided by the JSE on user registration.

Immediately after performing a TCP/IP connection, the client must send in a `TaxLogonReq`. The request contains for following information (mandatory fields in **bold** typeface):

Field Name	Description
member	User's member firm.
user	User identification. The user must belong to the member.
password	This is initially provided by the JSE but needs to be changed by the client on first log on. (see section 6.4.1).
ticket	Not used by JSE as pre-logon configuration is not enabled.
possDupSessId	If this session is replacing a previous session in a recovery scenario, this field should be set to the same value as the possDupSessionId of the previous session. See section 9.1.1.
majorVersion	See section 3.
minorVersion	See section 3.
microVersion	See section 3.

The response `TaxLogonRsp` contains following key information[1]:

Field Name	Description

[1] In addition to what is provided in all responses – see section 4.

logonAccepted	Indicates if the logon was successful.
loginStatus	• LOGIN_ACCEPTED: Set when (and only when) above logonAccepted is set to true. • WRONG_VERSION: The client and server support different EMAPI versions that are incompatible. • INITIAL_LOGIN: At initial logon, the password must be changed, see section 6.4.1. • PASSWORD_EXPIRED: The password has expired and must be changed, see section 6.4.1. • LOGIN_ACCESS_DENIED: The user does not have access to the logon service for this application. • LOGIN_REJECTED: Invalid Member/UserName/Password. • USER_ACCOUNT_LOCKED: The user is locked (for example, due to too many erroneous logon attempts or by the JSE). • USER_ACCOUNT_DISABLED: The user has been disabled by the JSE.
systemName	The name of RTC installation.
isTestSystem	Indication whether the installation is a test system or not.
clientHbtInterval	The suggested heartbeat interval. See section 6.3.
maxLostHeartbeats	The maximum number of hearbeat losses. See section 6.3.

Note: The TaxLogonReq should be sent immediately after the TCP connect, otherwise the session will be disconnected by RTC.

Note: A user can only have one connection at a time to the system. If the same user tries to log in again, the first connection will be closed. A system that needs to have several simultaneous connections to the system (for performance or resilience reasons, or for a functional partitioning of the user sessions) must have multiple user IDs with the same role.

6.3 Session Surveillance

The client is responsible for sending TaxHeartbeatReq after logging in. A suggested interval (in seconds) is specified in TaxLogonRsp (see section 6.2.1).

The request contains:

- userData: Client data that will be returned in the response.

The response TaxHeartbeatRsp contains the following information[2]:

- timestamp: Current local time in the server according to following ISO 8601 format: YYYY-MM-DDThh:mm:ss.sTZD (for example, 1997-07-16T19:20:30.45+01:00).
- userData: Client data specified in the request.

Note: Other EMAPI requests that are sent by the client do not replace the need to send TaxHeartbeatReq. Heartbeats must be sent to RTC TAX irrespective of whether there are

[2] In addition to what is provided in all responses – see section 5.1.

messages flowing through the connection or not. Heartbeats take priority over the transactional message request being submitted to the TAX server.

The server starts a heartbeat timer after a successful logon. The timer value is set to a value that is equal or greater than what is specified in `TaxLogonRsp`, i.e. a value greater than `maxLostHeartbeats` * `clientHbtInterval`. If this heartbeat timer expires, the server will terminate the session and the TCP connection.

The client must initiate a re-connection or failover when it does not receive a response to a `TaxHeartbeatReq` within a configurable time (should be configured in the client based on the `maxLostHeartbeats` setting in the TAX server).

Note: The server to fail over to will be provided by the JSE on user registration. Please see Section 9 for more details on recovery and failover procedures.

Note: The maximum retries for requests that fail in RTC will be determined by the JSE and communicated to clients on enablement.

6.4 Session Processing

This section contains session related information that is not part of the session establishment, surveillance, recovery or termination.

6.4.1 Change Password

A password may need to be changed for example when the `LoginStatus` field in `TaxLogonRsp` (see section 6.2.1) indicates `INITIAL_LOGIN` or when the password is about to expire.

To change a user's password, the client must submit a `ChangePasswordReq`. This request contains the following key fields:

Field Name	Description
`possDup`	The possible duplicate flag. See section 9.1.1
`member`	User's member firm.
`user`	User identification. The user must belong to the member.
`oldPassword`	The user's old password; used for authentication.
`newPassord`	The new password to be set.

The server returns a `ResponseMessage`[3] to a password change request.

Please note the following when changing passwords:

- password must contain at least one letter and one number
- password must be at 8 least characters long
- password is only valid for 30 calendar days
- after 3 failed attempts, RTC will lock out the account; clients will need to contact the JSE to unlock the account

Note: Passwords for the JSE will be managed by the JSE's authentication and authorisation system. The user does not need to be logged in to change the password.

[3] See section 5.1 for content.

6.4.2 Concurrent Processing of User Requests

It is configurable per RTC installation how many requests per user that the server processes in parallel. To make it possible to match the responses received from the server with the sent requests, the EMAPI header (See 4.1) contains a `clientTxRef` field. This field is set by the client in the header of the request and is returned by the server in the header of the response.

Note: When the server concurrently processes several requests from a user, the responses to the requests might not be received by the client in the same order as the client sent in the requests. For a client to correlate requests with responses, the client must set a unique `clientTxRef` for each request.

6.5 Session Termination

To terminate a session, submit a `TaxLogoutReq` to the server.

The response is a `ResponseMessage` and after that has been received the TCP connection will be disconnected.

If the session with the server is terminated due to some kind of failure, the client will not be able to send a `TaxLogoutReq`. Instead, the client will simply abandon the session and log on to the server as if it is a new connection being established.

If the session is terminated by the server, the last message published to the client is a `TaxSessionStatus` message, indicating the reason for the termination. This is just for information - the client does not need to act on this message.

6.6 Session Message Sequences

This section describes the sequence of messages when establishing and terminating a session as well as the required messages to keep the connection alive. For details on failover message sequences see section 9.

6.6.1 Session Establishment, Surveillance and Termination

The follow sequence diagram illustrates the messages for this scenario:

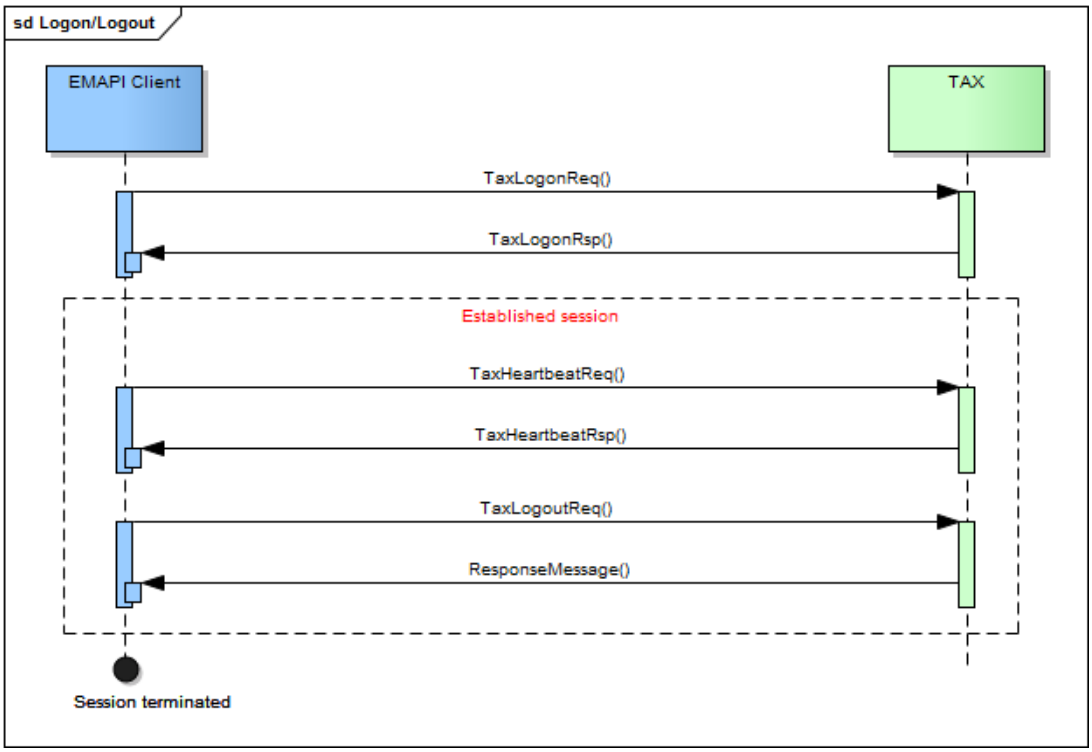

Figure 3 - Logon/Logout

Once a session is established, it must be maintained via heartbeats. `clientHbtInterval` specifies the interval (in seconds) that heartbeat requests must be sent.

Specify a unique `possDupSessId`. In case of failure, a new logon request will use the same `possDupSessId` enabling TAX to remove the duplicate session that failed.

Failure conditions:

- Malformed response.
- Correlation failure (`clientTxRef` returned is unknown).
- Error response (expected Response with error code or `ResponseMessage` or `simpleRsp`).
- Timeout.

Recovery

- Analyse error response.
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert.

7 SUBSCRIPTIONS

7.1 Broadcast Flows

Whenever a business event occurs, such a deal being allocated or collateral being pledged, RTC generates event messages. These event messages are grouped into a concept called Broadcast Flows. Each broadcast flow is configured with the event types that it will publish.

Note: The EMAPI broadcast flows are NOT to be confused with UDP-type multicast broadcasts. The events published on EMAPI broadcast flows are guaranteed as they are sent via the TCP/IP protocol and not UDP.

A particular event type will only be sent to one broadcast flow in RTC. The events that are published on the broadcast flows are described in more detail in *Volume 02 – Post-trade EMAPI Clearing (see 1.7)*.

A broadcast flow can support replay and/or current value. The following describes each subscription type:

Subscription Type	Description
Replay	It is possible to replay all earlier events that has happened since the RTC instance was started - see section 9.1.
Current Value	It is possible to receive the current state (for example, positions in an account). The current state may include terminated and inactive records, as long as they exist in RTC.

The list of broadcast flows that trading and clearing members can subscribe to is listed below:

Broadcast Flow	Current Value (Y/N)	Replay (Y/N)	Description and information sent
PUBLIC_GLOBAL_REFERENCE_DATA_FLOW	Yes	No[4]	Reference data events and Member Client events.
ACCOUNT_EVENT_FLOW	Yes	Yes	Positions and trades.
RISK_EVENT_FLOW	Yes	Yes	Risk calculation results.
GIVEUP_EVENT_FLOW	Yes	Yes	Assign and Tripartite workflow.
SETTLEMENT_EVENT_FLOW	Yes	Yes	Collateral and payment information.
MARKET_DATA_FLOW	Yes	No[5]	Market data events.

The broadcast flows do not go offline. The API user can connect and do replay as long as the system is up and running.

[4] To get missed data, issue a current value request to get lastest reference data.
[5] To get missed data, issue a current value request to get lastest market data.

7.2 Subscription Groups

Messages on flows from RTC are associated with a `SubscriptionGroup`. The subscription group has a numeric ID.

The subscription group is used to restrict the messages received on the flow. This restriction exists for two reasons:

- A subscriber may not be allowed to receive all information from the system, but is restricted to view one or more subscription groups.
- A subscriber may want to subscribe to less information than they are allowed to view, for performance reasons. This would be done by choosing to subscribe to specific subscription groups and not all subscription groups the subscriber is allowed to view.

7.2.1 Sequence Number

Each event on a flow that supports replay has a sequence number. Events on flows that only support current value and not replay are not sequenced. The sequence number is unique per:

- Broadcast Flow, and
- Subscription Group, if the flow is divided into different subscription groups.

The sequence numbers are positive integers and increases by 1; a gap in the sequence numbers should be treated as an error.

Events received as part of a current value (snapshot) when setting up a current value subscription, are assigned a dummy sequence number. Any real-time events following the snapshot will be sequenced. This only applies to flows that support current value and replay.

The reference data flow does not support replay, and its events are not sequence numbered.

7.2.2 Information in Events

All events on replayable flows contain the following information:

- `sequenceNumber`: See section 7.2.1.
- `subscriptionGroup`: Present in events sent on flows on which RTC sends events on different subscription groups. See section 7.3.1.

7.2.3 Finding the Latest Sequence Number

It is possible to find the latest published sequence number on a broadcast flow by using the `GetSequenceNumbersReq` request. A client specifies the broadcast flow and subscription group in the request. The response `GetSequenceNumbersRsp` contains the latest sequence number for this broadcast flow and subscription group. This can be useful to verify that there are no more messages that need to be recovered.

7.3 Subscription Establishment

When setting up a subscription or replay (using `TaxSnapshotSubscribeReq` or `TaxReplayReq`), the EMAPI client specifies the flow and subscription group. If the client wants to receive information from several subscription groups on that flow, one subscription must be set up for each subscription group. There may be several simultaneous subscriptions active on the same EMAPI session.

The system validates that the logged in user is allowed to view the information for the specified subscription group. If not, the subscription request is rejected. Once the subscription request has been accepted, there is normally no further access validation; every message for that subscription group is forwarded to the client.

To make it possible to match the events and framing messages received from the server with a subscription, the EMAPI header contains a `clientTxRef` field (see section 4.1) that is set by the client in the header of the subscription request and is returned by the server in the header of resulting messages (response, framing and event messages).

7.3.1 Finding the Right Subscription Group

An EMAPI client should subscribe to the `Reference Data Flow` to find their available subscription groups. A number of `SubscriptionGroup` reference data objects will be received, each with a numeric ID and a descriptive text.

Closely related to `SubscriptionGroup` are the AccessGroups. An `AccessGroup` is a way of limiting access on a member level. However, there will also be access groups related to the Clearing Member(CM) and Trading Member(TM) links. The `AccessGroup` reference data object also contains the subscription group ID for the link. The AccessGroup object makes it easier to identify the CM and TM, instead of interpreting the SubscriptionGroup's description text.

Example of `SubscriptionGroup` reference data received on the EMAPI reference data flow:

Field Name	Value
subscriptionGroupId	"1035"
description	"CM1_TM1"
accountAccessGroup	"CM1_TM1"

Example for `AccessGroup`:

Field Name	Value
accessGroupId	"CM1_TM1"
participantUnitId	"TM1"
clearingMemberId	"CM1"
subscriptionGroup	"1035"

Once an EMAPI client has identified the relevant subscription groups (received on the reference data flow), they should set up subscriptions to the desired subscription groups.

Messages on the reference data flow do not include a subscription group field; a client always receives all the reference data it is allowed to view. When setting up the reference data subscription, it is suggested that subscription group 1 be used in the `TaxSnapshotSubscribeReq`, although the value has no meaning for this flow.

7.3.2 Current Value Subscriptions

To get current value events and/or to subscribe to future events, the client sends a `TaxSnapshotSubscribeReq`. This message has the following key information:

Field Name	Description
flow	Data flow being requested. See section 7.1

key	The subscription group to subscribe to.
requestType	The client specifies one of following operations in the request: • CURRENT_VALUE: The server will only send events containing the current value (framed between TaxStartSnapshot and TaxEndSnapshot events). • SUBSCRIPTION: The server will send real-time events, but not the current value at that point in time. • CURRENT_VALUES_AND_SUBSCRIPTION: The server will send events containing the current value first (framed between TaxStartSnapshot and TaxEndSnapshot events) and then continue to send real-time events.
lastPollSequenceNumber	May only be set if requestType is CURRENT_VALUE. If the client only fetches current value at certain times the client shall set this field to the value of the pollSequenceNumber field of the TaxEndSnapshot (see below sub section) message previously received. If the current value hasn't changed since last time the client fetched the current value, no events will be received inside the current value framing. **Note:** RTC does not make use of this feature.
member	A user may be configured to be able to subscribe to private events that are sent to another member. To subscribe on behalf of the user shall set the member field in the TaxSnapshotSubscribeReq.

The response TaxSnapshotSubscribeRspcontains the following key fields[6]:

Field Name	Description
handle	Used when unsubscribing, see section 7.4.

Note: The TaxSnapshotSubscribeRsp is returned to the client by the server before any events are sent. The collection of the current value is done after the delivery of the TaxSnapshotSubscribeRsp, so events resulting from a request sent by the client after the reception of TaxSnapshotSubscribeRsp might be part of the current value.

Note: If a client has subscriptions (during the same time) that interleave, the client will receive some events that are identical (except possibly the clientTxRef field in the header).

[6] In addition to what is provided in all responses – see section 5.1.

Current value framing

A `TaxStartSnapshot` message precedes any current value events. This message contains the follow key information:

Field Name	Description
`flow`	See section 7.1
`subscriptionGroup`	See section 7.2.

A `TaxEndSnapshot` message follows any current value events. This message contains:

Field Name	Description
`code`	Set to `Ok` if the server has successfully delivered all events of the current value. All other values are indicating that the server has failed to deliver all events of the current value.
`message`	A text description explaining more in detail why the server has failed to deliver all events of the current value. **Note:** If the code is `Ok` the string will be “Ok”.
`subscriptionGroup`	See section 7.2.
`snapshotSize`	Number of events published in the snapshot.

7.3.3 Replay Subscriptions

To replay events—to recover events that have been sent by the server during times when the client hasn’t had any subscription—and optionally subscribe to future events, the client sends a `TaxReplayReq` (only for a flow that supports replay).

The `TaxReplayReq` request contains the follow key fields:

Field Name	Description
`flow`	See section 7.1
`subscriptionGroup`	See section 7.2.
`replayRequestType`	Specifies the type of replay. This could be either: • `REPLAY`: this is the default if this field isn’t set. For this type it is configurable per RTC installation how many events are allowed to be replayed. If the number of events to be delivered is greater than the configured number the server will only deliver this configured number of events and set the `nextSequence` field in the `TaxReplayEndEvent` message. The client must then initiate a new replay setting the `sequenceNumber` field in the `TaxReplayReq` message to the value of the field `nextSequence` in the `TaxReplayEndEvent` message received previously.

	• REPLAY_UNSEGMENTED: the server resends all events (framed according to the Replay Framing section) from (but exclusive) the specified sequenceNumber up to (and inclusive) the specified endSequenceNumber. • REPLAY_SUBSCRIPTION: the server resends all events (framed according to the Replay Framing section) from (but exclusive) the specified sequenceNumber up to the last sequence number sent by the server and then continues to send real-time events.
sequenceNumber	The sequence number from (but exclusive this number) where the replay will start. The client must specify a start sequence number that must either be: • 0 (zero) if the client hasn't received any events, for example, at client start-up. • The sequence number of the last received event—when the client has received events earlier, for example, if the client is recovering after a failover. • The value of nextSequence field in TaxReplayEndEvent if the replay events are delivered in several replays. Only applicable if the replay request type is REPLAY. See also section 7.2.1.
endSequenceNumber	May only be set if replayRequestType is REPLAY or REPLAY_UNSEGMENTED. The sequence number to which the replay will be done. If not set, the server will replay all events that it knows of at the time the replay is finished.
member	A user may be configured to be able to subscribe to private events that are sent to another member. To subscribe on behalf of the user shall set the member field in the TaxReplayReq.

The response TaxReplayRsp[7] contains the following additional key field:

- handle: Used when unsubscribing, see section 7.4.

Note: The TaxReplayRsp is returned to the client by the server before any events are sent. The collection of the replayed events is done after the delivery of TaxReplayRsp, so events resulting from a request sent by the client after the reception of TaxReplayRsp might be part of the replayed events.

Note: If a client performs several replays (during the same time) that interleave, the client will receive some events that are identical (except possibly the clientTxRef field in the header).

Replay framing

A TaxReplayStartEvent message precedes any events that are resent. This message contains:

Field Name	Description

[7] In addition to what is provided in all responses – see section 5.1.

flow	See section 7.1
subscriptionGroup	The subscription group to replay from/subscribe to. See section 7.2

A TaxReplayEndEvent message follows any replayed events. This message contains:

Field Name	Description
code	Set to Ok if the server has successfully replayed all events. All other values are indicating that the server has failed to replay all events.
message	A text description explaining more in detail why the server has failed to replay all events. **Note:** If the code is Ok the string will be "Ok".
subscriptionGroup	The subscription group to replay from/subscribe to. See section 7.2
nextSequence	Only applicable for REPLAY request type. See the description of the replayRequestType field in TaxReplayReq and the section 7.3.4 Synchronise Subscription/Replay.

Restrictions

It is recommended to only replay from zero at client start-up and during extreme client situations, for example, client is restarted clean. That is, if a client is required to perform replay(s) it must:

- Remember the last received sequence number per subscription.
- Set up subscriptions for flows that it may be required to perform replay(s) as early as possible.

See also section 7.3.3 on replay subscriptions.

7.3.4 Synchronise Subscription/Replay

The client can separately perform replay(s) and subscription establishment to get the same result as using the REPLAY_SUBSCRIPTION replay request type. In order to do that the client should:

- replay events using TaxReplayReq with replayRequestType set to REPLAY/REPLAY_UNSEGMENTED
- set up a subscription using TaxSnapshotSubscribeReq with requestType set to SUBSCRIPTION

To avoid handling too much live and replayed data at the same time, the client is advised to:

1. Replay events until the nextSequence field isn't set in the TaxReplayEndEvent message.
2. Setup the subscription.

7.4 Subscription Termination

To terminate a subscription `TaxRemoveSubscriptionReq` must be sent to the server. This message contains the following additional key field:

- `handle`: The subscription handle received in `TaxSnapshotSubscribeRsp` or in `TaxReplayRsp`.

7.5 Subscription Message Sequences

This section contains message sequence examples.

7.5.1 Snapshot Subscriptions

The following sequence diagram illustrates the messages between a client and the TAX server when requesting a snapshot of Account Event Flow data.

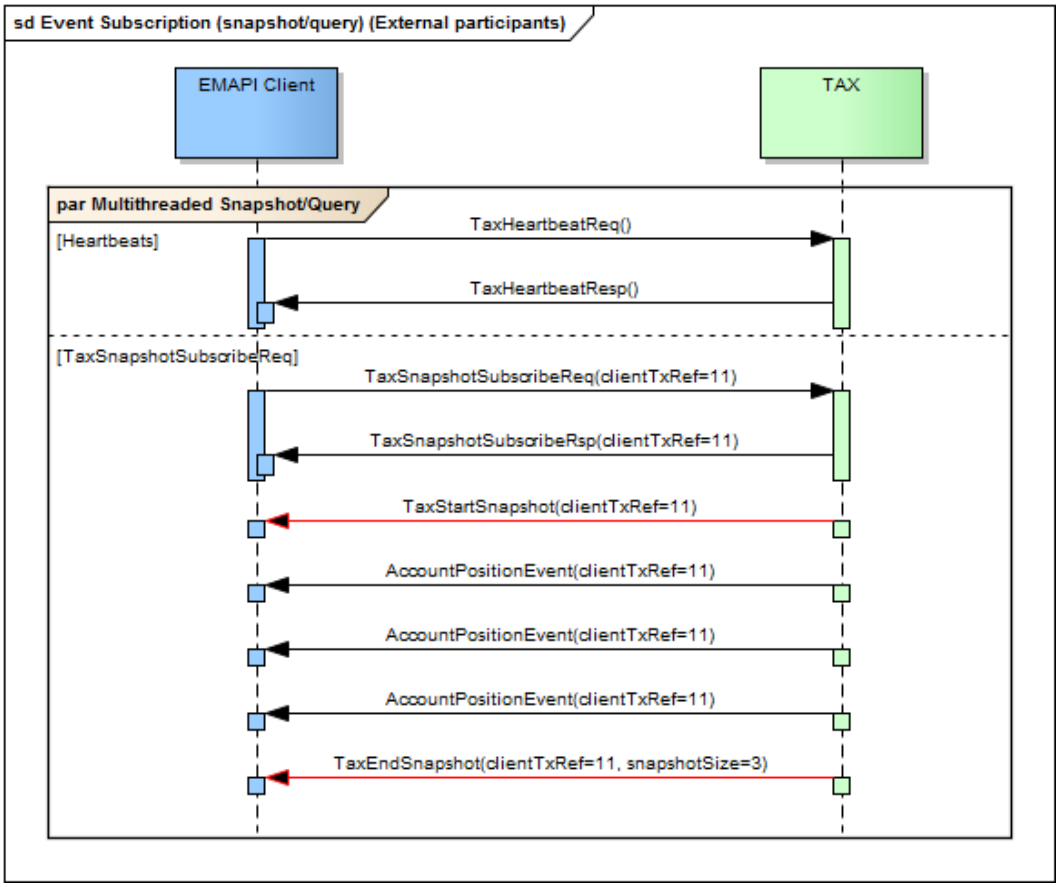

Figure 4 - Event Subscription (snapshot)

Key fields on `TaxSnapshotSubscribeReq`:

Field Name	Description
`SubscriptionRequestType`	1 = CURRENT_VALUE
`Flow`	301 = ACCOUNT_EVENT_FLOW
`key`	6 = (example key)

Failure conditions for: `TaxSnapshotSubscribeReq/TaxSnapshotSubscribeRsp`

- Malformed response.
- Correlation failure (`clientTxRef` returned is unknown).
- Error response (expected Response with error code or `ResponseMessage` or `SimpleRsp`).
- Timeout.

Recovery

- Analyse error response
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert

Note: `possDup` does not have to be set for resending a `TaxSnapshotSubscribeReq`, as a snapshot/query does not update the state of the system. It will be ignored.

Failure conditions for: `TaxStartSnapshot` **/** `AccountPositionEvent` **/** `TaxEndSnapshot`

- Malformed messages.
- Correlation failure on `AccountPositionEvents` (unknown `clientTxRef`).
- Timeout waiting for `TaxStartSnapshot`.
- `TaxEndSnapshot` not received.
- `TaxEndSnapshot snapshotSize` does not match the number of events received in the snapshot/query.

Recovery

- Analyse error response.
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Terminate the session (`TaxLogoutReq` followed by `ResponseMessage`) and reinitiate the session - i.e. logon (`TaxLogonReq` followed by `TaxLogonRsp`), and resend the request.
- Halt and alert.

7.5.2 Replay Subscriptions

The following diagram illustrates the flow of messages for subscription replay:

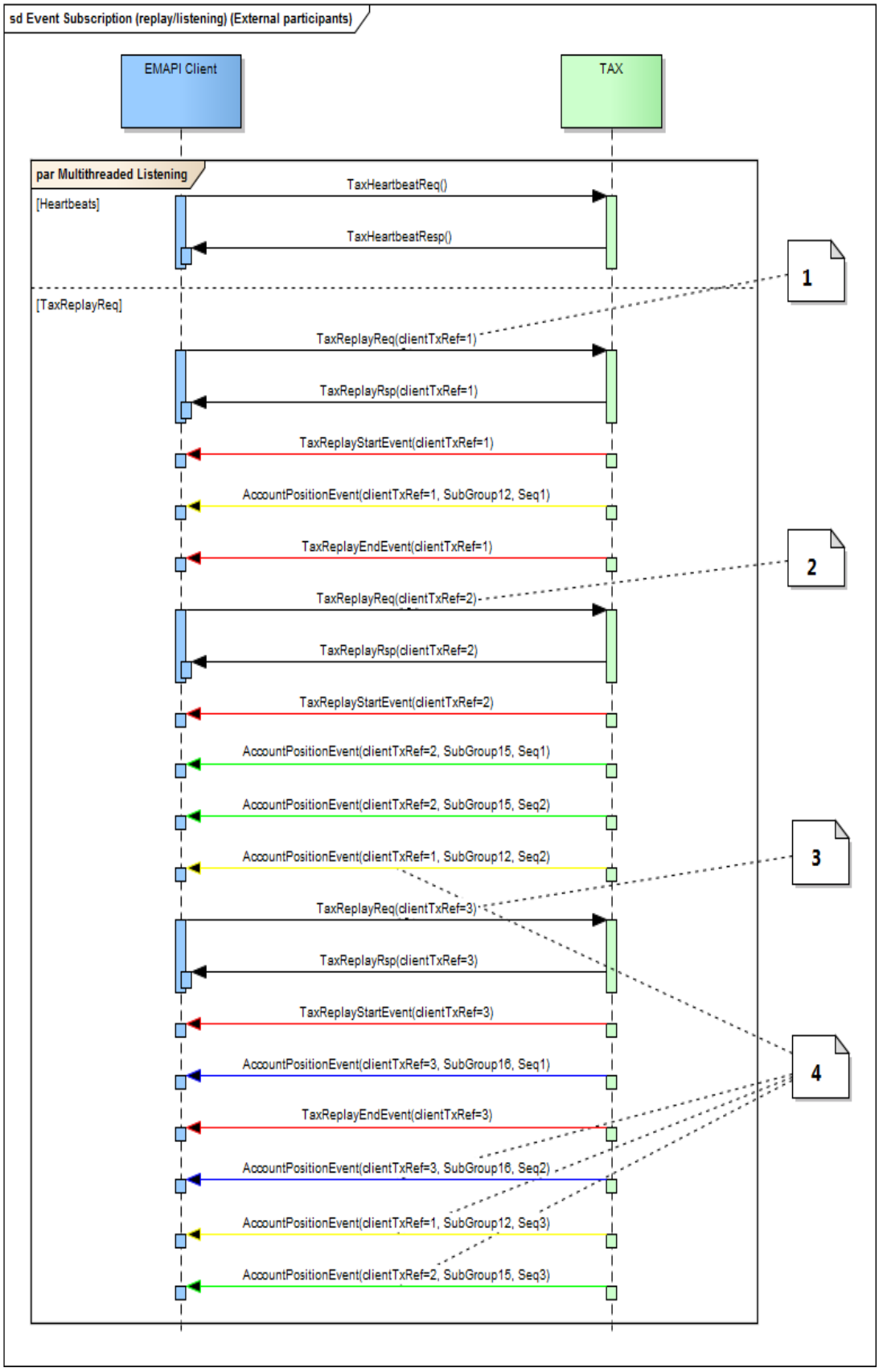

Figure 5 - Replay Subscription

Label 1:

`TaxReplyReq(clientTxRef=1)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	12 = (example subscription group)
`sequenceNumber`	0

Label 2:

`TaxReplyReq(clientTxRef=2)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	15 = (example subscription group)
`sequenceNumber`	0

Label 3:

`TaxReplyReq (clientTxRef=3)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	16 = (example subscription group)
`sequenceNumber`	0

Label 4:

These are "live" events that are sent after the `TaxReplayEndEvent` is sent.

Failure and Recovery Conditions:

a) `TaxReplayStartEvent/AccountPositionEvent/TaxReplayEndEvent`

Failure conditions:

- Malformed messages

- Correlation failure on `AccountPositionEvent` (unknown `clientTxRef`)
- Timeout waiting for `TaxReplayStartEvent`
- `TaxReplayEndEvent` not received
- Sequence gap (e.g. SubGroup1 Seq3 followed by SubGroup1 Seq5)

Recovery

- Analyse error response
- Resend request (`TaxReplayStartEvent`) (with `possDup=true` and original `clientTxRef`) or
- Halt and alert.

Recovery (Sequence Gap)

- Terminate replay subscription (`TaxRemoveSubscriptionReq`)
- Reinitiate the replay subscription; with sequenceNumber = <last successful sequence for the subscription group with a sequence gap>.

b) `TaxReplayReq/TaxReplayRsp`

Failure conditions:

- Malformed response
- Correlation failure (`clientTxRef` returned is unknown)
- Error response (expected Response with error code or `ResponseMessage` or `SimpleRsp`)
- Timeout.

Recovery:

- Analyse error response
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert.

7.6 Building a copy of the reference data cache

All reference data objects published on the Reference Data Flow contain a set of fields that can be used by an EMAPI client to build its own copy of the reference data. An EMAPI client should examine the ”action” field to determine how the reference data message should affect the cache. A client can subscribe to an initial snapshot and then apply subsequent updates (add/update/remove) in order to build an up-to-date copy of the cache.

7.6.1 CACHE_ACTION

The field "action" specifies if the reference data object is added, updated or removed. All objects published in a reference data snapshot will have the action set to "add". See the documentation for the `CACHE_ACTION` enumeration (in the EMAPI message description in HTML) to find which values should be treated as add, update or remove.

Added reference data objects:

If the action is set to ADD(1) or BOOTLOAD(3), the reference data object has been added to the cache. All objects published in a reference data snapshot will have the action set to ADD.

Updated reference data objects:

If the action is set to UPDATE(2), the reference data object has been updated. The client should replace the old copy of the object in its cache with the new one.

Removed reference data objects:

If the action is set to any of the four values (4-7) that indicate removal, the client should remove the reference data object from its cache. RTC will not publish any further updates for this reference data object. EMAPI clients should treat all four types of removal in the same way. After a reference data object has been removed, it will not be included in the reference data snapshot (since the snapshot only contains currently active reference data objects).

Note: For `Instrument` and `TradableInstrument`, a message is published with the `IsEnabled` flag set to `FALSE` on rollover of the business day following the `ValidToDate`. No message with `CACHE-ACTION` of remove (values 4 to 7) will be published. After business rollover, the `Instrument` and `TradableInstrument` will no longer be available as part of a snapshot subscription.

7.6.2 Other fields

The field "`cacheId`" specifies which of the internal reference data structures in RTC that the object belongs to. This information is not relevant to an EMAPI client and can be ignored.

The field "`uniqueObjectId`", if set, is a unique identifier for each reference data object. The field "key" is the uniqueObjectId for the object's parent object, if the object belongs to a tree structure. If the object is the root node of a tree, this field has a null value.

The field "`stateSequenceNumber`" (SSN) is a sequence number for the reference data. Every update to RTC's reference data increases the `stateSequenceNumber`.

An EMAPI client should check the SSN of the incoming data on the reference data flow to the updates of the local cache copy in the correct order.

8 RECONCILIATION

EMAPI clients can query the latest sequence number per flow and subscription group using the `GetSequenceNumbersReq` message in order to perform end of day reconciliation between their systems and RTC. The following diagram illustrates the flow of messages for this scenario:

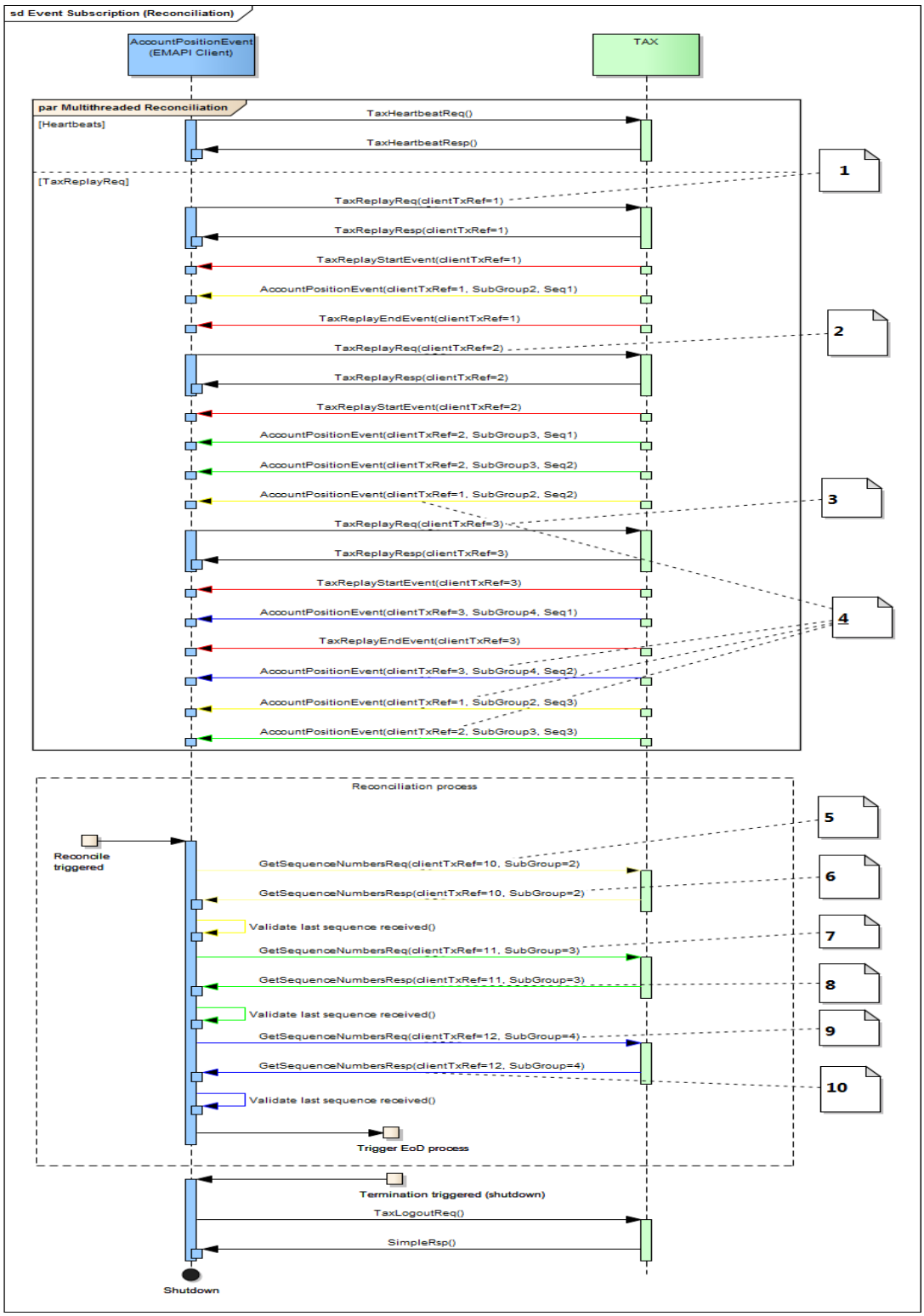

Figure 6 - Reconciliation

Label 1:

`TaxReplyReq(clientTxRef=1)` has the following values:

Field Name	Description
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	2 = (example subscription group)
`sequenceNumber`	0

Label 2:

`TaxReplyReq(clientTxRef=2)` has the following values:

Field Name	Description
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	3 = (example subscription group)
`sequenceNumber`	0

Label 3:

`TaxReplyReq(clientTxRef=3)` has the following values:

Field Name	Description
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	4 = (example subscription group)
`sequenceNumber`	0

Label 4:

These are "live" events that are sent after the `TaxReplayEndEvent` is sent.

Label 5:

`GetSequenceNumbersReq(clientTxRef=10)` has the following value:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	2 = (example subscription group)

Label 6:

`GetSequenceNumbersRsp(clientTxRef=10)` has the following values:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	2 = (example subscription group)
`sequenceNumber`	3

Label 7:

`GetSequenceNumbersReq(clientTxRef=11)` has the following values:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	3 = (example subscription group)

Label 8:

`GetSequenceNumbersRsp(clientTxRef=11)` has the following values:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	2 = (example subscription group)
`sequenceNumber`	3

Label 9:

`GetSequenceNumbersReq(clientTxRef=12)` has the following values:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	4 = (example subscription group)

Label 10:

`GetSequenceNumbersRsp(clientTxRef=12)` has the following values:

Field Name	Description
`broadcastFlowId`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroupId`	4= (example subscription group)
`sequenceNumber`	2

Failure conditions and Recovery

a) GetSequenceNumbersReq / GetSequenceNumbersRsp

Failure Conditions

- Malformed response
- Correlation failure (`clientTxRef` returned is unknown)
- Error response (expected Response with error code or `ResponseMessage` or `SimpleRsp`)
- Timeout.

Recovery

- Analyse error response
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert.

b) Last sequence received

Failure conditions:

- `sequenceNumber` not equal to last message's sequence number.

Recovery

- If `sequenceNumber` is less than last message's sequence number (i.e. JSE has more than what was produced) then halt and alert
- If `sequenceNumber` is greater than last message's sequence number, then send `TaxReplayReq` with relevant `SubscriptionGroup` and `sequenceNumber` of missing message.

9 RECOVERY AND FAILOVER

9.1 Session Recovery

The session establishment at failover is done in the same way as initial establishment (see section 6.2). In the case of disaster recovery, the client should connect to recovery site. The connectivity details to the failover site are provided to clients on enablement.

9.1.1 Outstanding Requests

An outstanding request is a request that the client has sent but has not received any response for yet. The recommendation for a maximum response timeout is 5 seconds.

A session may fail after the client has sent in some request(s) but before the client has received the response(s) to these requests.

A `possDup` field is available in some messages to indicate if the message is being sent to RTC as a possible duplicate. In such a case, the client must resend the concerned outstanding requests with the `possDup` (=possible duplicate) field set.

If the `possDup` field is not present in the message, the client must still resend the outstanding message. For these messages, RTC will perform a duplicate check.

If a request is found which is equal to the resent request with the `possDup` field set, the server returns the response either immediately if the request has already been completely served or when the ongoing request has been completely served. That is, the server will answer with the same response independently of if the server has received or hasn't received the request before.

In some cases, the response code will indicate that a request had been processed before. The client should treat these response codes in the same way as an `Ok`, since the request was successfully processed the first time. The response codes that should be treated as `Ok` are:

- `RtcSettlement_MESSAGE_ALREADY_PROCESSED`
- `RtcClearing_INVALID_CLIENT_DEAL_ID`
- `RtcClearing_DUPLICATE_MOVE_TRADE`
- `RtcClearing_FOUR_EYES_ALREADY_EXISTS`
- `RtcClearing_INVALID_FOUR_EYES_STATE`
- `RtcClearing_INVALID_STATUS`
- `RtcMessageAlreadyProcessed`

The client must only set the `possDup` field on requests for which the client has not received any response on an earlier session establishment. This is due to that the possible duplicate checking in the server adds latency to the transaction.

The number of requests the server keeps for a member and user combination to be used for possible duplicate checking is limited. This will be provided by the JSE upon user registration. A client shall never send in more outstanding requests than this number.

Note: `possDup=true` signals RTC to check the message body in case it's a duplicate, i.e. it does not pertain to the header fields. Therefore, the `clientTxRef` field does not have to be the same if the message is being resent with possDup=true.

Note: Please see Appendix B for details on how to handle outstanding requests for each message.

The following diagram illustrates the flow of messages for a recovery scenario:

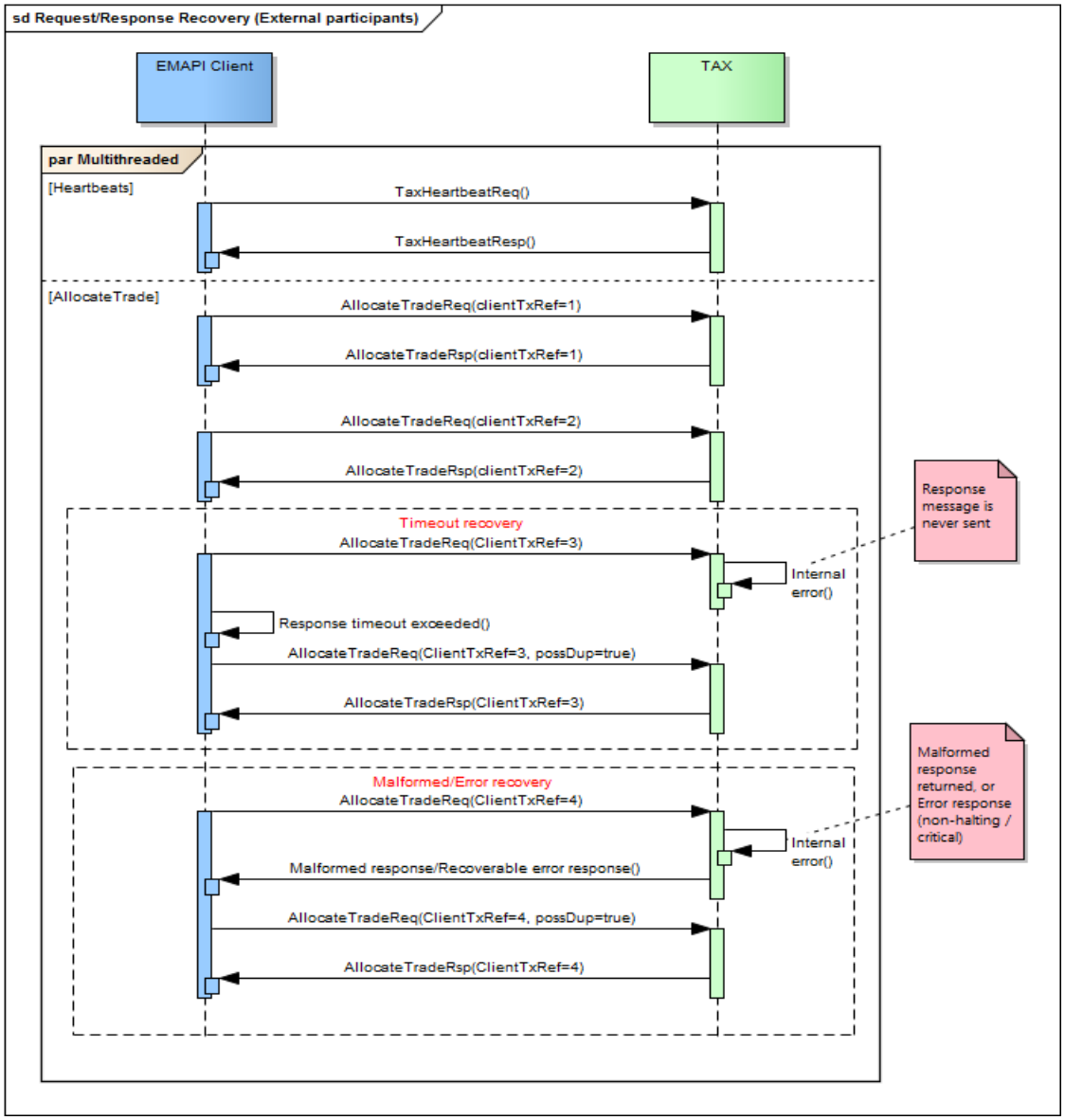

Figure 7 - Request/Response Recovery

9.2 Subscription Recovery

9.2.1 Current Value Subscriptions

After discarding earlier received events, the subscription establishment at failover is done in the same way as initial establishment - see section 7.3.2

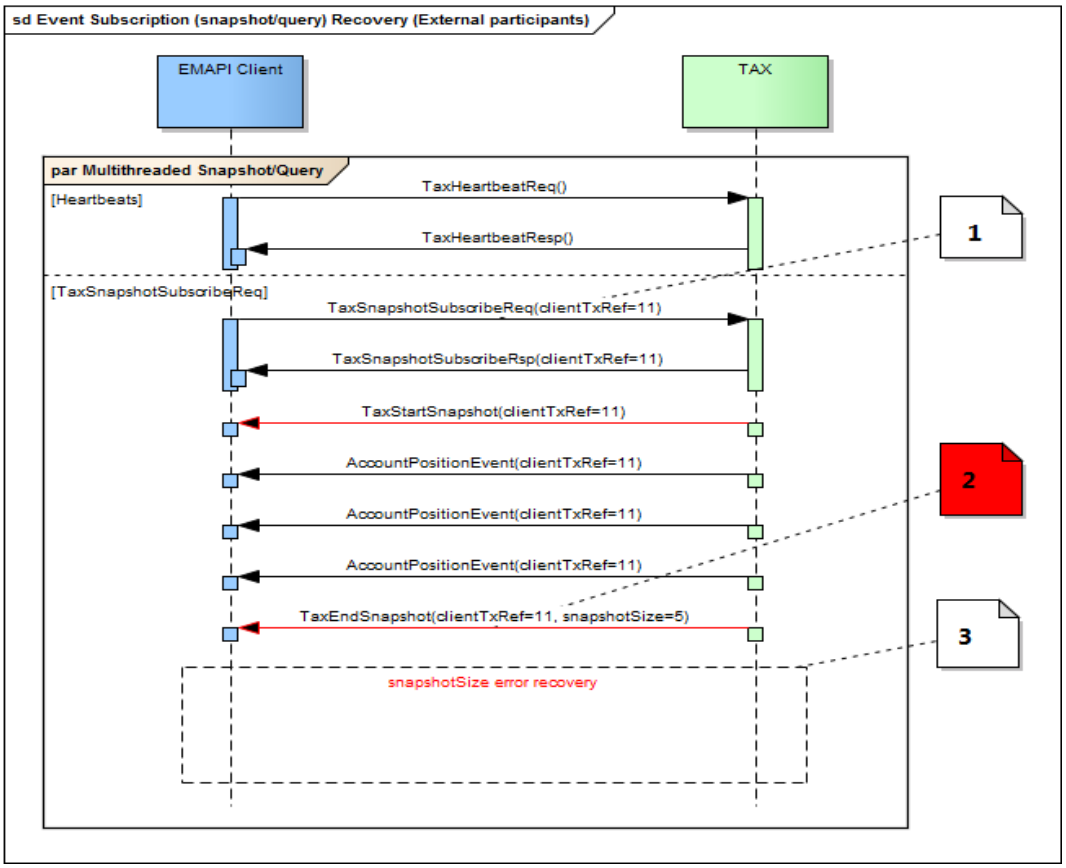

Figure 8 - Current Value Subscriptions

Label 1:

`TaxSnapshotSubscribeReq` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	6 = (example subscription group)

Label 2:

- **Error 1** - `snapshotSize`=3 expected, but 5 received, or
- **Error 2** - Timeout waiting for `TaxEndSnapshot` message

Label 3:

- **Repeat** `TaxSnapshotSubscribeReq` as in Label 1.

Failure conditions for: `TaxSnapshotSubscribeReq/ TaxSnapshotSubscribeRsp`

- Malformed response
- Correlation failure (`clientTxRef` returned is unknown)
- Error response (expected Response with error code or `ResponseMessage` or `SimpleRsp`)
- Timeout.

Recovery

- Analyse error response
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and Alert.

Note: `possDup` does not have to be set for resending a `TaxSnapshotSubscribeReq`, as a snapshot/query does not update the state of the system. It will be ignored.

Failure condition for: `TaxStartSnapshot`**/**`AccountPositionEvent`**/** `TaxEndSnapshot`

- Malformed messages
- Correlation failure on `AccountPositionEvents` (unknown `clientTxRef`)
- Timeout waiting for `TaxStartSnapshot`
- `TaxEndSnapshot` not received
- `TaxEndSnapshot snapshotSize` does not match the number of events received in the snapshot/query.

Recovery

- Analyse error response
- Resend request (with possDup=true and original `clientTxRef`) or,
- Terminate the session (`TaxLogoutReq` followed by `ResponseMessage`) and reinitiate the session - i.e. logon (`TaxLogonReq` followed by `TaxLogonRsp`), and resend the request
- Halt and alert.

9.2.2 Replay Subscriptions

The subscription establishment at failover is done in the same way as initial establishment (see section 7.3.3) with the difference that the earlier last received sequence number is specified in the request[8].

The following sequence diagram illustrates the flow of messages for this scenario:

[8] Zero is specified at subscription establishment.

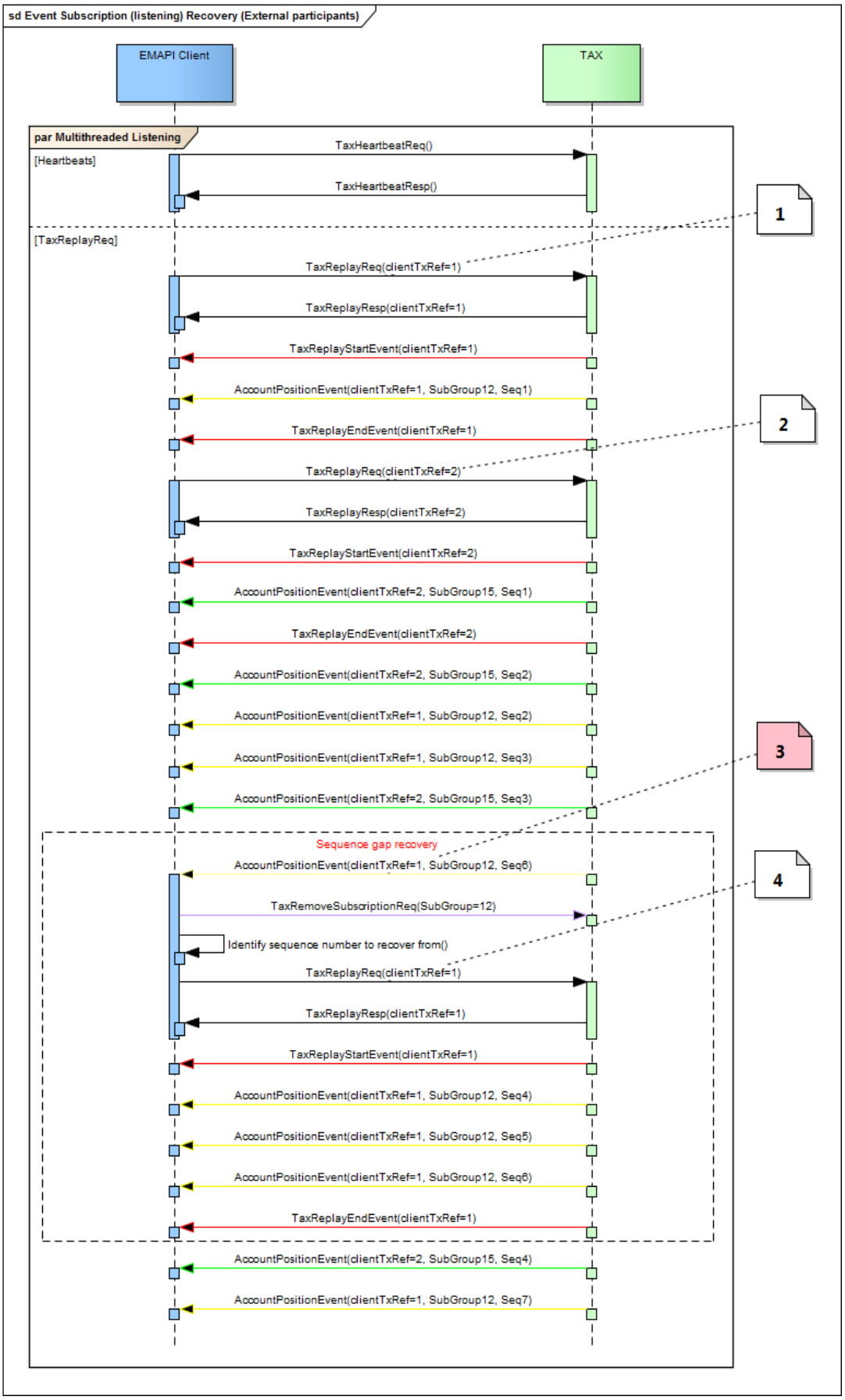

Figure 9 - Event subscription recovery

Label 1:

`TaxReplyReq(clientTxRef=1)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	12 = (example subscription group)
`sequenceNumber`	0

Label 2:

`TaxReplyReq(clientTxRef=2)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	15 = (example subscription group)
`sequenceNumber`	0

Label 3

Error - Sequence 4 expected, but 6 received 3

Label 4

`TaxReplyReq(clientTxRef=1)` has the following values:

Field Name	Description
`requestType`	2 = 'SUBSCRIPTION'
`flow`	301 = ACCOUNT_EVENT_FLOW
`subscriptionGroup`	12 = (example subscription group)
`sequenceNumber`	4

Failure conditions - for `TaxReplayReq/TaxReplayRsp`

- Malformed response
- Correlation failure (`clientTxRef` returned is unknown)
- Error response (expected Response with error code or `ResponseMessage` or `SimpleRsp`)
- Timeout.

Recovery

- Analyse error response
- Resend request (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert.

Failure conditions - for the `TaxReplayStartEvent / AccountPositionEvent / TaxReplayEndEvent`

- Malformed messages
- Correlation failure on `AccountPositionEvent` (unknown `clientTxRef`)
- Timeout waiting for `TaxReplayStartEvent`
- `TaxReplayEndEvent` not received
- Sequence gap (e.g. SubGroup12 Seq3 followed by SubGroup12 Seq5).

Recovery

- Analyse error response
- Resend request (`TaxReplayStartEvent`) (with `possDup=true` and original `clientTxRef`) or,
- Halt and alert.

Recovery (Sequence Gap)

- Terminate replay subscription (`TaxRemoveSubscriptionReq`)
- Client reinitiates the replay subscription, with `sequenceNumber` = <last successful sequence for the subscription group with a sequence gap>.

9.2.3 Failover

The JSE will provide clients with details regarding disaster recovery failover servers such as IP addresses. The following diagram illustrates the flow of messages for a failover scenario:

Note: The JSE is still working on the final aspects of Disaster Recovery; this will be confirmed in due course once the final design is finalised.

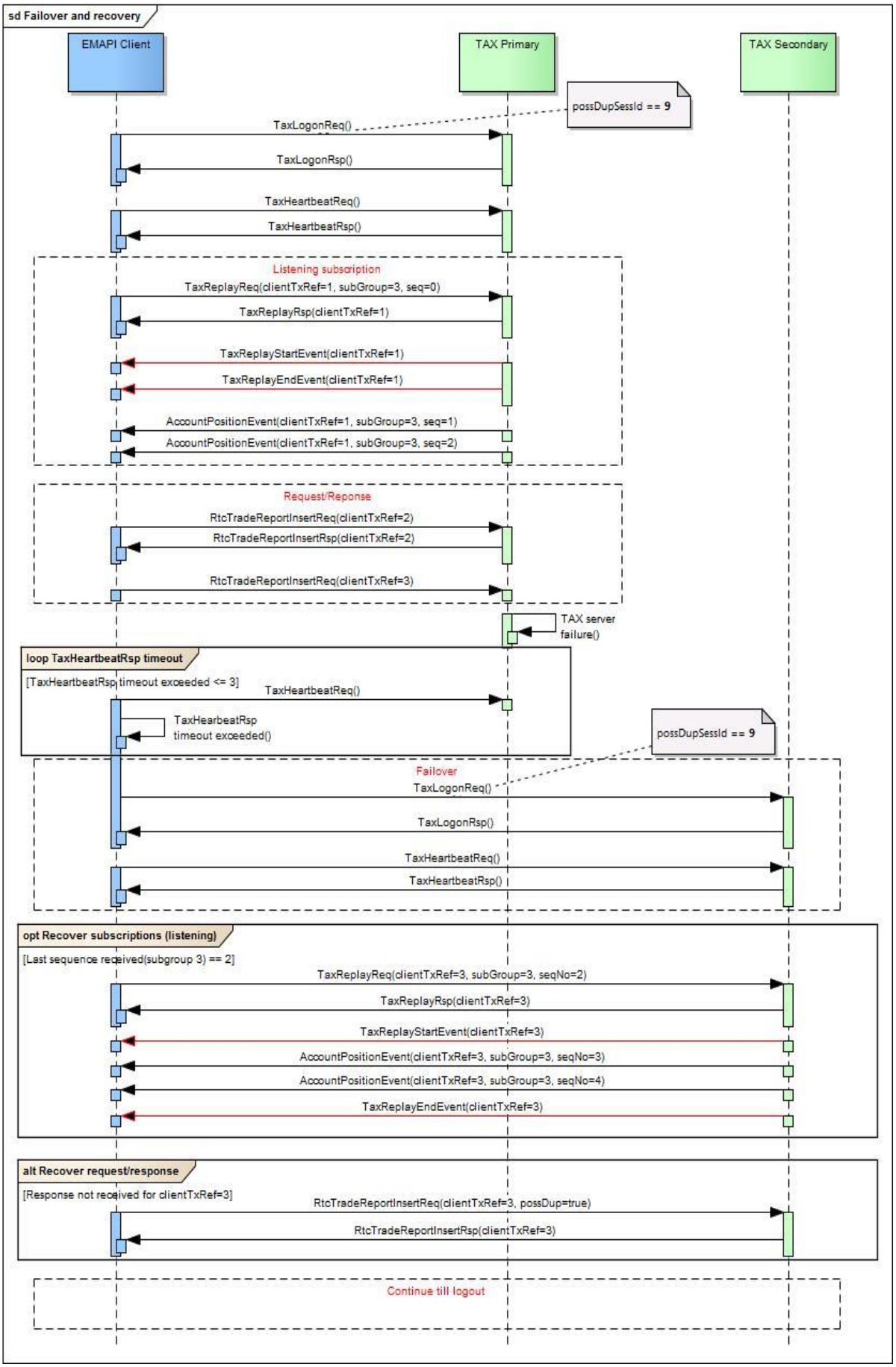

Figure 10 - Failover and recovery

APPENDIX A – MESSAGE FORMATS

Note: This Appendix contains the common or administration messages in *EmapiTransactionsForMember.html* for ease of reference in the document via hyperlinks. Please refer to the complete set of technical specification documents published on the ITaC website: https://www.jse.co.za/services/itac.

Note: The message definitions in this appendix always reflect the most recent version of EMAPI - for a view of what has changed since the last update, please refer to the *EmapiTransactionsRevHistForMember* file(s) published on the ITAC Website under the *EMAPI Revision History Versions* section.

Version 1.36.1 (Build: not released)

General Messages

General Messages (Internal)

Reference Data Messages

Message: CdResponse

Message: CdResponse

Message ID: 227

Type: General Messages

Description: A response to be used as super class for all responses from CD.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	latestSSN		long	This is the latest and most likely the highest state sequence number, SSN, that has been assigned to the reference data.

Message: ChangePasswordReq

Message ID: 126

Type: General Messages

Description: A request to change the current password. The user does not have to be logged in in order to change the password.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	possDup		boolean	The possible duplicate flag
12	memberId		String	The id of the user's member (firm). Required because usernames are only unique within a member firm.
13	userId		String	The identification of the user (username).
14	oldPassword		String	The user's old password, used for authentication.
15	newPassword		String	The new password to be set.

This request will normally return a response of type CdResponse .

Message: GetSequenceNumbersReq

Message ID: 10430

Type: General Messages

Description: Get sequence numbers for broadcast flows.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
6	broadcastFlowId	required	int	Broadcast Flow requested.
7	subscriptionGroupId	required	int	Request sequence number for this subscription group.

This request will normally return a response of type GetSequenceNumbersRsp .

Message: GetSequenceNumbersRsp

Message ID: 10431

Type: General Messages

Description: Response to a GetSequenceNumbersReq request.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
6	sequenceNumber		long	Latest sequence number for the requested broadcast flow and subscription group.
7	broadcastFlowId		int	Broadcast Flow.
8	subscriptionGroupId		int	Subscription group.

Message: ResponseMessage

Message ID: 230

Type: General Messages

Description: General response for request messages that dont't have a defined response. It may also be used when a fatal error occurs before or during the normal response handling on the server.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	messageReference		String	The message reference from the corresponding RequestMessage.

Message: SimpleRsp

Message ID: 231

Type: General Messages

Description: General response for request messages that dont't have a defined response.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment

1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	reply		String	Generic single string reply

Message: TaxEndSnapshot

Message ID: 73

Type: General Messages

Description: Message ending a snapshot response

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	The overall status
2	message		String	Complementary text for extra context-dependent info
3	subCode		int []	Subcodes
4	flow		Integer	If this message is the result of a snapshot/subscribe operation on a flow then this field contains the flow id.
5	pollSequenceNumber		Long	Not used in this configuration of RTC.
6	subscriptionGroup		Integer	Identifying group of instruments in a current value response if applicable, otherwise zero
10008	snapshotSize		Long	Number of items published in snapshot

---

Message: TaxHeartbeatReq

Message ID: 75

Type: General Messages

Description: Heartbeat sent to gateway in order to verify a connection

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
2	userData		String	User supplied data. The data is returned in the response.

This request will normally return a response of type TaxHeartbeatRsp .

Message: TaxHeartbeatRsp

Message ID: 76

Type: General Messages

Description: Response returned from gateway

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	reply		String	Generic single string reply
6	timestamp		String	Current central system time. The format is "yyyy-MM-ddTHH:mm:ss.SSS". Example: 2009-07-16T19:20:30.045
7	userData		String	User-supplied data from the request

Message: TaxLogonReq

Message ID: 63

Type: General Messages

Description: Request to the gateway to log in a member/user

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
2	member	required	String	User's member firm
3	user	required	String	Mandatory user id. The user must belong to the member.
4	password	required	String	User's password
5	ticket		Long	Ticket received at pre-login
6	possDupSessId		Integer	Possible duplicate session id. If two sessions (that is, users) have the same possDupSessId it means that an unacknowledged request on one of the sessions can be resent on the other with the possDup flag set and the system will be able to resolve if it is a duplicate or not. Not used in this configuration of RTC.
7	majorVersion		int	EMAPI major version. If any of the version fields is non-zero, the gateway will validate against the current EMAPI version.
8	minorVersion		int	EMAPI minor version. If any of the version fields is non-zero, the gateway will validate against the current EMAPI version.
9	microVersion		int	EMAPI micro version. If any of the version fields is non-zero, the gateway will validate against the current EMAPI version.

This request will normally return a response of type TaxLogonRsp .

Message: TaxLogonRsp

Message ID: 64

Type: General Messages

Description: Sent from the gateway to the client as a response to TaxLogonReq.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment

1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	reply		String	Generic single string reply
6	logonAccepted		Boolean	Indicates whether the login was successful or not.
7	loginStatus		int	Login specific status code.
8	isTestSystem		Boolean	Indicates whether this system is a test system or not.
9	systemName		String	The name of the system.
10	partitionHbtInterval		Integer	The interval (in seconds) between partition heartbeats sent from the system. Partition heartbeats are sent out as Heartbeat events.
11	clientHbtInterval		Integer	The interval (in seconds) between which clients are expected to send in heartbeats. The client should use the TaxHeartbeatReq message to send in heartbeats.
12	maxLostHeartbeats		Integer	The maximum number of heartbeats to lose before the connection can be considered to be down.

Message: TaxLogoutReq

Message ID: 65

Type: General Messages

Description: Request from client to gateway in end a session. A simple response is sent as response.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment

This request will normally return a response of type SimpleRsp .

Message ID: 71

Type: General Messages

Description: Removes an active subscription. A SimpleRsp is sent as response for this request.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
2	handle		int	Subscription handle (subscription identifier) identifying the subscription request to be removed. The handle is received in the response when setting up the subscription.

This request will normally return a response of type SimpleRsp .

Message: TaxReplayEndEvent

Message ID: 235

Type: General Messages

Description: Framing message indicating the end of requested replay data. The TaxReplayEndEvent indicates the end of a replay sequence.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	subscriptionGroup		int	The subscription group the data is for. The identifier is always set to zero for global flows.
2	nextSequence		Long	When requesting a replay, the trading system may not deliver the full sequence in the first call. The application may need to issue multiple additional requests for retrieving all data. The field "nextSequence" indicates if all data has been retrieved. If so, the field is NULL. Otherwise, the field indicates the sequence number to be used when requesting the next/following batch of replay data.
3	statusCode		int	EMAPI status code telling if the replay was successful or not.
4	statusMessage		String	Status text associated with the EMAPI status code returned.
5	internalCode		int	Not used in this configuration of RTC.
6	flow		int	The flow the data is for.

Message: TaxReplayReq

Message ID: 232

Type: General Messages

Description: Request message sent to the RTC system to recover a sequence of messages published earlier. The replay request will recover earlier published messages on a replayable flow. The response back is a simple response indicating whatever the request was successfully queued to the RTC system. The actual replay data is delivered as unsolicited events, framed by TaxReplayStartEvent and TaxReplayEndEvent messages.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
2	flow		int	Specifies the logical stream of information of a certain type. Allowed values: see constant group BroadcastFlows
3	subscriptionGroup		Integer	The subscription group on the subscribed flow.
4	sequenceNumber		long	The sequence number from which messages should be recovered for the specified subscription group and flow.
6	member		String	Optional attribute defining the member for which the replay is to be applied for. Used for on-behalf-of replay. Note that the user requesting replay for another member must be authorized to do so. If this attribute is left empty, the logged in user's member is used.
7	endSequenceNumber		long	The sequence number up to which messages should be recovered for the specified subscription group and flow. The value for this attribute could be derived from the TaxSnapshotSubscribeRsp.
8	requestType		int	The type of replay request. Allowed values: see constant group ReplayRequestType
10009	segmentSize		Integer	Not used in this configuration of RTC.

This request will normally return a response of type TaxReplayRsp .

Message: TaxReplayRsp

Message ID: 233

Type: General Messages

Description: Response message sent back for a previously-submitted TaxReplayReq. The TaxReplayRsp response will not contain the actual data being requested. The response data is delivered to the application asynchronously.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	reply		String	Generic single string reply
6	handle		int	Subscription handle identifying the subscription request. The handle is used when removing the subscription.

Message: TaxReplayStartEvent

Message ID: 234

Type: General Messages

Description: Framing message indicating the start sequence of requested replay data. When issuing a replay request, the replay data is delivered as unsolicited messages. The TaxReplayStartEvent indicates the start of a replay sequence.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	subscriptionGroup		int	The subscription group the data is for.
2	flow		int	The broadcast flow for the start event.

Message: TaxSessionStatus

Message ID: 77

Type: General Messages

Description: Unsolicited message indicating session status.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	status		int	Session status Allowed values: see constant group SessionStatus

Message: TaxSnapshotSubscribeReq

Message ID: 69

Type: General Messages

Description: Request to retrieve information and/or activate subscription of future updates of the information specified

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
2	member		String	Not used in this configuration of RTC.
3	user		String	Not used in this configuration of RTC.
4	requestType		int	Type of subscription request Allowed values: see constant group SubscriptionRequestType
5	flow		int	Data flow being requested Allowed values: see constant group BroadcastFlows
6	key		int	Selection key, identifying the data being subscribed to. In many cases, this is the subscription group.
7	sequenceNumber		long	Not used in this configuration of RTC.
8	lastPollSequenceNumber		Long	Not used in this configuration of RTC.

This request will normally return a response of type TaxSnapshotSubscribeRsp .

Message: TaxSnapshotSubscribeRsp

Message ID: 70

Type: General Messages

Description: Response to a subscription request (TaxSnapshotSubscribeReq).

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	code		int	Status code. Code 3001 indicates that the request was processed successfully. For other codes, see the Status Code list in the EMAPI HTML description.
2	message		String	A textual description of the status code above.
3	subCode		int []	Status code for each leg of the request. Only used for batched requests.
5	reply		String	Generic single string reply
6	handle		int	Subscription handle identifying the subscription request. The handle is used when removing the subscription.
7	lastPublishedSeqNo		Long	Not used in this configuration of RTC.

Message: TaxStartSnapshot

Message ID: 72

Type: General Messages

Description: Message preceding a snapshot response

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	subscriptionGroup		Integer	Group of instruments in current value response if applicable, otherwise zero.
2	flow		int	The broadcast flow for the start event

General Messages (Internal)

Message: CdRequest

Message ID: 226

Type: General Messages

This message can only appear as a sub-object in other messages; it can never be used as a stand-alone message.

Description: This message is not used in EMAPI.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	possDup		boolean	The possible duplicate flag
2	member		String (64)	The member of the *user*.
3	user		String (64)	The responsible user. • Shall only be set in requests if *actingUser* is acting on behalf of another user. • Otherwise TRADExpress will set it to the *actingUser*.
6	actingUser		String (64)	The user that initiated a request. • Shall only be set in requests if (and only if) it is a gateway user that sends the request. • Otherwise TRADExpress will set it to the user that established the session on which the request was received on. See also the *user* field.

Message: RequestMessage

Message ID: 237

Type: General Messages

This message can only appear as a sub-object in other messages; it can never be used as a stand-alone message.

Description: This message is not used in EMAPI

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	possDup		boolean	The possible duplicate flag

Reference Data Messages

Message: AccessGroup

Message ID: 10051

Type: Reference Data Messages

Description: This object defines an access Group.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	key		String	Is used to identify the parent object (is set to null if this is the root object). This field is set by RTC, only set on outgoing messages on the reference data flow.
2	cacheId		String	Is used to identify the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow.
3	action		Integer	Identify the reason for the cache action (CACHE_ACTION), i.e. if it is an addition of a new reference data object, an update of an existing object or a removal of an object from the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow. Allowed values: see constant group CACHE_ACTION
4	stateSequenceNumber		long	A sequence number that is incremented with each reference data update, i.e. a version number for the cache contents. The sequence number series is common for all caches. This means that for a specific cache instance, the sequence number is not necessarily consecutive (but constantly increasing). This field is set by RTC, only set on outgoing messages on the reference data flow.
5	uniqueObjectId		String	The id is unique among all objects and may be used to retrieve a specific instance. Do not however, try to interpret the contents. This field is set by RTC, only set on outgoing messages on the reference data flow.
6	timeStamp		String (23)	The date and time of the latest modification for this reference data object. Format: yyyy-mm-ddTHH.MM.SS.sss. May be null if the object never has been updated. This field is set by RTC, only set on outgoing messages on the reference data flow.
8	accessGroupId		String	The id for the Access Group.
9	participantUnitId		String	Specifies the parent Participant Unit.
10	clearingMemberId	required	String	The clearing member for the Access group.
11	subscriptionGroup		int	The subscription group for the Access group.

Message: Member

Message ID: 101

Type: Reference Data Messages

Description: This object represents a member firm and holds all basic member data such as id, full name, mail addresses and contact persons etc.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	key		String	Is used to identify the parent object (is set to null if this is the root object). This field is set by RTC, only set on outgoing messages on the reference data flow.
2	cacheId		String	Is used to identify the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow.
3	Action		Integer	Identify the reason for the cache action (CACHE_ACTION), i.e. if it is an addition of a new reference data object, an update of an existing object or a removal of an object from the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow. Allowed values: see constant group CACHE_ACTION
4	stateSequenceNumber		long	A sequence number that is incremented with each reference data update, i.e. a version number for the cache contents. The sequence number series is common for all caches. This means that for a specific cache instance, the sequence number is not necessarily consecutive (but constantly increasing). This field is set by RTC, only set on outgoing messages on the reference data flow.
5	uniqueObjectId		String	The id is unique among all objects and may be used to retrieve a specific instance. Do not however, try to interpret the contents. This field is set by RTC, only set on outgoing messages on the reference data flow.
6	timeStamp		String (23)	The date and time of the latest modification for this reference data object. Format: yyyy-mm-ddTHH.MM.SS.sss. May be null if the object never has been updated. This field is set by RTC, only set on outgoing messages on the reference data flow.
7	memberId	required	String (64)	The public ID of the participant. This id has to be unique.
8	fullName	required	String (128)	Complete name of the member firm
10	address		String (255)	Company postal address.
11	Phone		String (64)	Company phone number.
12	Fax		String (64)	Company fax address.
13	complianceContact		String (128)	Name of contact person in compliance matters
14	complianceContactPhone		String (64)	Phone number to the compliance contact person
15	complianceContactMail		String (128)	Mail address to the compliance contact person
16	matchingContact		String (128)	Matching/BackOffice contact person
17	matchingContactPhone		String (64)	Phone number to the Matching/BackOffice contact person
18	matchingContactMail		String (128)	Mail address to Matching/BackOffice contact person
19	isDisabled		Boolean	Set to true if this member has been disabled.

26	associatedMemberId		String (255)	The parent member, for example the parent of a trading member branch or a client.
30	memberType		Integer	The type of member. N.B. in the documentation member type is also known as Participant type. Allowed values: see constant group MemberType
33	validFromDate		String (10)	The date from which the member is valid. The format is yyyy-MM-dd
34	listOfAliases		String (256)	A list of other markets'/exchanges' ID of this member in the format: market1:memberid1,market2:memberid2,... "maket1" is assumed to be a market defined in the "local" system and is automatically converted to uppercase, since this is the conversion for market ids in the system.
39	allowedOnBehalfOfMemberIdList		String (255)	A comma separated list of Member Ids for which this member may act on behalf of.
10043	participantUnitType		Integer	The type of this participant unit. Allowed values: see constant group ParticipantUnitType
10046	isStaff		Boolean	Is Staff, true or false
10047	isBeneficial		Boolean	Is Beneficial, true or false
10048	allowClientSubAccounts		Boolean	Sub accounts are allowed, true or false
10049	vatRegNumber		String	VAT registration number
10050	bdaCode		Integer	Broker Deal Account number
10051	Email		String	Email
10052	country		String	Country Code, e.g. ZA
10053	isNonResident		Boolean	Is Non Resident is true if country for the client is not equal to ZA
10054	nominatedMember		String	MemberId of the nominated member, fulfilling physical settlement when the actual member is not in the Equities market. Otherwise, set to the member itself.
10055	strateCode		String (100)	Code of client or member at CSD.
10056	externalPayment		Boolean	If RTC should expect net payment from the JSE integration layer for this member. Required for clearing members.
10057	ownTM		String	This is set for clearing members only, to indicate its own trading member.
10058	allowFxCollateral		Boolean	This field is mandatory for CMs. If true, RTC will expect system-to-system communication on size of FX collateral payments.
10059	allowedMarkets		String	A comma-delimited list of market codes that the client is allowed to have trades and positions in.
10060	waitForCmBalancing		Boolean	This field is mandatory for CMs. If true, RTC will expect the CM to send in a response to a balancing event.
10061	clientType		String	For clients only - type of client. Information to surveillance. Required for all clients. Allowed values: see constant group ClientType
10062	idNumber		String	For clients only - ID number. Required for local individual clients: Client Type = Individual AND isNonResident = FALSE

10063	passportNumber		String	For clients only - Passport number. Required for foreign individual clients: Client Type = Individual AND isNonResident = TRUE
10064	companyRegistrationNumber		String	For clients only - Company registration number. Required for all company clients: Client Type = Company
10065	isProfessional		Boolean	For clients only - Information to surveillance. Required for all clients.
10066	isShariah		Boolean	For clients only - Information to surveillance. Required for all clients.
10067	isDiscretionary		Boolean	Is Discretionary, true or false
10068	preferredCcy		String (3)	Currency used for intraday rebalancing. Needs to be an eligible FX collateral currency (or ZAR, this will be the default though).
10069	branchMemberNumber		String	For TM branches only. Unique within a TM. Valid number is between 01 and 99.
10070	cmMessageRef		String (5)	For CM only. Mandatory for CM. Number used when creating settlement instructions. This number is concatenated into the message reference no.

Message: SubscriptionGroup

Message ID: 96

Type: Reference Data Messages

Description: The subscription group is used to filter objects on broadcast flows. When a subscription is set up for a subscription group the system controls the user access rights for that access group.

Field no.	Field name (tag)	Mand.	Type (max length)	Comment
1	key		String	Is used to identify the parent object (is set to null if this is the root object). This field is set by RTC, only set on outgoing messages on the reference data flow.
2	cacheId		String	Is used to identify the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow.
3	action		Integer	Identify the reason for the cache action (CACHE_ACTION), i.e. if it is an addition of a new reference data object, an update of an existing object or a removal of an object from the reference data cache. This field is set by RTC, only set on outgoing messages on the reference data flow. Allowed values: see constant group CACHE_ACTION
4	stateSequenceNumber		long	A sequence number that is incremented with each reference data update, i.e. a version number for the cache contents. The sequence number series is common for all caches. This means that for a specific cache instance, the sequence number is not necessarily consecutive (but constantly increasing). This field is set by RTC, only set on outgoing messages on the reference data flow.
5	uniqueObjectId		String	The id is unique among all objects and may be used to retrieve a specific instance. Do not however, try to interpret the contents. This field is set by RTC, only set on outgoing messages on the reference data flow.
6	timeStamp		String (23)	The date and time of the latest modification for this reference data object. Format: yyyy-mm-ddTHH.MM.SS.sss. May be null if the object never has been updated. This field is set by

				RTC, only set on outgoing messages on the reference data flow.
7	subscriptionGroupId	required	Integer	The business id of the subscription group.
8	description	required	String (255)	A text description of the set of order books contained in the group.
9	partitionId	required	Integer	The partition this subscription group belong to.
10012	accountAccessGroup		String	The account access group.

Messages by ID

ID	Message name
63	TaxLogonReq
64	TaxLogonRsp
65	TaxLogoutReq
69	TaxSnapshotSubscribeReq
70	TaxSnapshotSubscribeRsp
71	TaxRemoveSubscriptionReq
72	TaxStartSnapshot
73	TaxEndSnapshot
75	TaxHeartbeatReq
76	TaxHeartbeatRsp
77	TaxSessionStatus
96	SubscriptionGroup
101	Member
126	ChangePasswordReq
226	CdRequest
227	CdResponse
230	ResponseMessage
231	SimpleRsp
232	TaxReplayReq
233	TaxReplayRsp
234	TaxReplayStartEvent
235	TaxReplayEndEvent
237	RequestMessage
10051	AccessGroup

Constants

BroadcastFlows
CACHE_ACTION
DIVISOR
LoginStatus
MemberType
ReplayRequestType
RtcState
SchedulerState
SessionStatus
SubscriptionRequestType

Constant group: BroadcastFlows
Description: Defines broadcast flows

Constant name	Type	Value	Comment
PUBLIC_GLOBAL_REFERENCE_DATA_FLOW	int	11	Global reference data flow.
ACCOUNT_EVENT_FLOW	int	301	Account event flow.
RISK_EVENT_FLOW	int	302	Risk event flow.
MARKETDATA_EVENT_FLOW	int	303	Market Data event flow.
GIVEUP_EVENT_FLOW	int	304	GiveUp event flow.

Constant group: CACHE_ACTION
Description: Defined cache actions

Constant name	Type	Value	Comment
ADD	int	1	Add to cache EMAPI - interpret as Add
UPDATE	int	2	Update cache EMAPI - interpret as Update
BOOTLOAD	int	3	Add to cache with bootloader EMAPI - interpret as Add
REMOVE_CACHE_DB	int	4	Remove from cache and db, does not remove if there are references to object. Return status code ValidationHasReference if referenced. EMAPI - interpret as Remove
REMOVE_CACHE_DB_FORCED	int	5	Remove from cache and db, removes even if there are references to object. EMAPI - interpret as Remove
REMOVE_CACHE	int	6	Remove from cache (does not remove object from db), does not remove if there are references to object. Return status code ValidationHasReference if referenced. The isDeleted attribute is set to BOOLEAN.TRUE EMAPI - interpret as Remove
REMOVE_CACHE_FORCED	int	7	Remove from cache (does not remove object from db), removes even if there are references to object. The isDeleted attribute is set to BOOLEAN.TRUE EMAPI - interpret as Remove

Constant group: DIVISOR
Description: There are integer/long fields that represent decimal numbers. These need to be divided with the following constants.

Constant name	Type	Value	Comment
QTY	int	1000000	Divisor for quantity field
PRICE	int	1000000	Divisor for price fields
INTEREST	int	1000000	Divisor for interest fields.
DELTA	int	1000000	Divisor for delta fields.
DECIMAL	int	1000000	Divisor for decimal value fields.

Constant group: LoginStatus
Description: Provides the result of a login request.

Constant name	Type	Value	Comment
LOGIN_ACCEPTED	int	0	The login is accepted.
LOGIN_REJECTED	int	-1	The login is rejected due to invalid password or invalid user id.
USER_ACCOUNT_LOCKED	int	-2	User account is locked due to too many erroneous login attempts.
PASSWORD_EXPIRED	int	-3	The password has expired.
LOGIN_ACCESS_DENIED	int	-4	User does not have access to login service for this application.
WRONG_VERSION	int	-5	Client and TAX server versions are not compatible.
INITIAL_LOGIN	int	-6	Initial login, password must be changed.
USER_ACCOUNT_DISABLED	int	-7	Account disabled by operational staff.

Constant group: MemberType
Description: Defines the different member/participant types.

Constant name	Type	Value	Comment
MARKETPLACE	Integer	1	The Clearing House itself.
MEMBER_UNIT	Integer	7	A member unit is a type of member that must be connected to a parent member, for example to divide an organization into different departments. Trading Member Branches and Clients are both of the type MEMBER_UNIT.

CLEARING_ONLY_MEMBER	Integer	8	A Clearing Member.
TRADING_ONLY_MEMBER	Integer	9	A Trading Member.

Constant group: ParticipantUnitType
Description: Participant type. Defines the type of participant a member has in the member tree.

Constant name	Type	Value	Comment
CLEARING_MEMBER	Integer	1	Clearing Member.
TRADING_MEMBER	Integer	2	Trading Member
CLIENT	Integer	3	Client.
TRADING_MEMBER_BRANCH	Integer	4	Trading Member Branch.

Constant group: ReplayRequestType
Description: Literals describing the type of replay request

Constant name	Type	Value	Comment
REPLAY	int	0	Request to replay specific events; no future updates
REPLAY_UNSEGMENTED	int	1	Request to replay specific events without having to issue requests for new segments
REPLAY_SUBSCRIPTION	int	2	Request for unsegmented replay of events up to the latest and for subsequent subscription to future updates

Constant group: RtcState
Description: System state in RTC.

Constant name	Type	Value	Comment
OPEN	String	"OPEN"	Open.
END_OF_TRADE_MANAGEMENT	String	"END_OF_TRADE_MANAGEMENT"	Trade management is no longer allowed.
END_OF_DAY	String	"END_OF_DAY"	End of Day process started.
POST_END_OF_DAY	String	"POST_END_OF_DAY"	End of Day process completed.

Constant group: SessionStatus
Description: Session status

Constant name	Type	Value	Comment
FORCED_LOGOFF_BY_NEW_LOGIN	int	1	The session has been terminated due a new login with the same user.
FORCED_LOGOFF_USER_DISABLED	int	2	The session has been terminated because the user has been disabled.
FORCED_LOGOFF_USER_DELETED	int	3	The session has been terminated because the user has been deleted.
FORCED_LOGOFF	int	4	User session logout was forced. Caused by an operator terminating the session.
DISCONNECT	int	5	User session disconnected
NORMAL_LOGOFF	int	6	Normal user requested logout

Constant group: SchedulerState
Description: State of the scheduler.

Constant name	Type	Value	Comment
NORMAL	Integer	1	Normal state during daily operations.
RERUN_EOD	Integer	2	This state is used during End of Day rerun.
INTRADAY_MARGIN_CALL	Integer	3	This current system state is used during Intraday Margin Call.
REBALANCING	Integer	4	This current system state is used during Intraday Collateral Rebalancing.

Constant group: SubscriptionRequestType
Description: Literals describing the type of subscription request

Constant name	Type	Value	Comment
CURRENT_VALUE	int	1	Request for current values only; no future updates

SUBSCRIPTION	int	2	Request for subscription of future updates only; no current value
CURRENT_VALUES_AND_SUBSCRIPTION	int	3	Request for current values and future updates

APPENDIX B – MESSAGE RESUBMITION

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
63	TaxLogonReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
65	TaxLogoutReq	Client needs to re-submit message using same value as per original message	Error message will be raised. This should NOT be treated as OK.	Message processed
69	TaxSnapshotSubscribeReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
71	TaxRemoveSubscriptionReq	Client needs to re-submit message using same value as per original message	Error message will be raised. This should NOT be treated as OK.	Message processed
75	TaxHeartbeatReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
126	ChangePasswordReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
226	CdRequest	Message not used	N/A	N/A
232	TaxReplayReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
237	RequestMessage	Message not used	N/A	N/A
10031	CdAddRtcMemberClientReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10034	CdAddRtcPositionAccountReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10049	AggregateTradesReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as ‘OK’ if part of the set of error messages for duplicates.	Message processed
10072	CdEnableDisableRtcPositionAccountReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10104	AllocateTradeReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10108	CorrectAllocationErrorReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10110	CorrectPrincipalReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10112	ModifyPositionSubAccountReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10114	AssignTradeReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10127	CdAddRtcMemberClientClearingLinkReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10128	CancelGiveUpReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10130	ApproveGiveUpReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10132	RejectGiveUpReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10134	TripartiteAllocationReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10147	CdUpdateRtcMemberClientReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10148	ModifyTradeSubAccountReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10152	CdEnableDisableRtcMemberClientReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10186	ExerciseOptionPositionReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as ‘OK’ if part of the set of error messages for duplicates.	Message processed
10188	AbandonOptionPositionReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as ‘OK’ if part of the set of error messages for duplicates.	Message processed
10258	QueryTradesReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
10267	CdAddCashAccountReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10268	CdUpdateCashAccountReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10273	CdSetMinimumZARLimitReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10293	CdSetTradingMemberRiskLimitReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10294	CdSetClientRiskLimitReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10384	GetRequestsForFXCollateralReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
10386	RegisterFXCollateralReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10420	SetCmBalancingStatusReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
10442	CdSetTradingMemberAMPercentageReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10443	CdSetClientAMPercentageReq	Message with PossDup = TRUE	RTC response 'OK'.	Message processed
10487	ConfirmWithdrawalsReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10491	GetPaymentAdvicesReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
10515	AddCommissionReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10516	CancelCommissionReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10517	RejectCommissionReq	Client needs to re-submit message using same value as per original message	Client must treat 'Error Message' (as per Section 9.1.1 in Volume PT01) as 'OK' if part of the set of error messages for duplicates.	Message processed
10527	QueryDividendPaymentFactorsReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message processed
10270	GetRiskArrayReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message re-processed

ID	Message name	User Action	RTC Response (Duplicate)	RTC Response (Not Duplicate)
10301	GetSettlementInstructionsReq	Client needs to re-submit message using same value as per original message	Message re-processed	Message re-processed