Elektron Message API Java Edition V3.0

ELEKTRON MESSAGE API CONFIGURATION GUIDE

Document Version: 1.0
Date of issue: 29 April 2016
Document ID: EMAJ300CG.160

THOMSON REUTERS

Contents

Chapter 1 Introduction

1.1 About this Manual

This document is authored by Elektron Message API architects and programmers. Several of its authors have designed, developed, and maintained the Elektron Message API product and other Thomson Reuters products which leverage it. As such, this document is concise and addresses realistic scenarios and use cases.

This guide documents the functionality and capabilities of the Elektron Message API Java Edition . The Elektron Message API can also connect to and leverage many different Thomson Reuters and customer components. If you want the Elektron Message API to interact with other components, consult that specific component's documentation to determine the best way to configure and interact with these other devices.

This document explains the configuration parameters for the Elektron Messaging API (simply called the Message API). Message API configuration is specified first via compiled-in configuration values, then via an optional user-provided XML configuration file, and finally via programmatic changes introduced via the software.

Configuration works in the same fashion across all platforms.

1.2 Audience

This manual provides information that aids software developers and local site administrators in understanding Elektron Message API configuration parameters. You can obtain further information from the *Elektron Message API Developer's Guide*.

1.3 About Message API Configuration

You write the Message API configuration using a simple XML schema, some settings of which can be changed via software function calls. The initial configuration compiled into the Message API software defines a minimal set of configuration parameters. Message API users can also supply an XML file (**EmaConfig.xml**) to specify configuration parameters. Additionally, programmatic interfaces can change parameter settings.

Message API configuration data is divided into four types:

- ***Consumer***: Consumer configuration data is the highest-level description of the application. Such settings typically select entries from the channel and dictionary groups.
- ***Channel***: Channel configuration data describe various connection alternatives and provides configuration alternatives for those connections.
- ***Dictionary***: Dictionary configuration data sets the location information for dictionary alternatives.

This manual discusses the four configuration groups and the configuration parameters available to each group.

1.4 Acronyms and Abbreviations

ACRONYM	MEANING
ADH	Advanced Data Hub
ADS	Advanced Distribution Server

Table 1: Acronyms and Abbreviations

ACRONYM	MEANING
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
EDF	Elektron Data Feeds
EED	Elektron Edge Device
EMA	Elektron Message API, referred to simply as the Message API
ETA	Elektron Transport API, referred to simply as the Transport API
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol (Secure)
OMM	Open Message Model
QoS	Quality of Service
RDF Direct	Reuters Data Feed Direct
RDM	Reuters Domain Model
RMTES	Reuters Multi-Lingual Text Encoding Standard
RSSL	Reuters Source Sink Library
RWF	Reuters Wire Format
TREP	Thomson Reuters Enterprise Platform
UML	Unified Modeling Language
UTF-8	8-bit Unicode Transformation Format

Table 1: Acronyms and Abbreviations

1.5 References

1. *Elektron Message API Java Edition RDM Usage Guide*
2. *API Concepts Guide*
3. *Elektron Message API Developers Guide*
4. *Transport API Java Edition Developers Guide*
5. specific to the programming language you use.

1.6 Documentation Feedback

While we make every effort to ensure the documentation is accurate and up-to-date, if you notice any errors, or would like to see more details on a particular topic, you have the following options:

- Send us your comments via email at apidocumentation@thomsonreuters.com.
- Add your comments to the PDF using Adobe's **Comment** feature. After adding your comments, submit the entire PDF to Thomson Reuters by clicking **Send File** in the **File** menu. Use the apidocumentation@thomsonreuters.com address.

1.7 Document Conventions

This document uses the following types of conventions:

- Typographic
- Field and Text Values

1.7.1 Typographic

- Java classes, methods, in-line code snippets, and types are shown in **`orange, Courier New`** font.
- Parameters, filenames, tools, utilities, and directories are shown in **Bold** font.
- Document titles and variable values are shown in *italics*.
- When initially introduced, concepts are shown in ***Bold, Italics***.
- Longer code examples are shown in Courier New font against an orange background. For example:

```
AppClient client = new AppClient();

OmmConsumerConfig config = EmaFactory.createOmmConsumerConfig();

OmmConsumer consumer  =
        EmaFactory.createOmmConsumer(config.operationModel(OperationModel.USER_DISPATCH)
        .host("localhost:14002").username("user"));

ReqMsg reqMsg = EmaFactory.createReqMsg();

consumer.registerClient(reqMsg.domainType(EmaRdm.MMT_MARKET_BY_PRICE).serviceName
        ("DIRECT_FEED").name("BBH.ITC"), client);
```

1.7.2 Field and Text Values

The value for individual fields in XML files are specified as **<*fieldName* value="*field_value*"/>** where:

- ***fieldName*** is the name of the field and cannot contain white space.
- ***field_value*** sets the field's value and is always included in double quotes.

> **Note:** Except for examples, double quotes are omitted from the field (parameter) descriptions throughout the remainder of this document.

Though enumerations have text values (i.e., SOCKET), in the software, text values are represented as numbers (required for programmatic configuration). When introduced, enumerations are listed along with their textual values.

Chapter 2 EMA Global Configuration Parameters

2.1 Parameter Overview

Many default behaviors are hard-coded into the EMA library and globally enforced. However, if you need to change EMA behaviors or configure EMA for your specific deployment, you can use EMA's XML configuration file (**EmaConfig.xml**) and adjust behaviors using the appropriate parameters (discussed in this section). While EMA globally enforces a set of default behaviors, certain other default behaviors are dependent on the use of the XML file and its settings.

For a list of default behaviors (and the parameters that you can use to change these behaviors) refer to Section 2.2.

For details on editing **EmaConfig.xml** and its XML schema, refer to Chapter 2, EMA Global Configuration Parameters.

2.2 Default Behaviors

When the EMA library needs a parameter, it behaves according to its hard coded configuration. You can change the behavior of EMA by providing a valid alternate value either through the use of **EmaConfig.xml**, function calls, or programmatic methods.

PARAMETER	TYPE	DEFAULT BEHAVIOR	NOTES
Host	String	localhost	Specifies the host name of the server to which the application connects. The parameter value can be a remote host name or IP address.
Port	String	14002	Specifies the port number on the server to which the application connects.
DefaultConsumer	String	EmaConsumer	If consumer components are configured, this parameter is ignored.
RdmFieldDictionaryFileName	String	./RDMFieldDictionary	Specifies the path and name of the **RdmFieldDictionary** file.
EnumTypeDefFileName	String	./enumtype.def	Specifies the path and name of the **enumtypeDef** dictionary file.

Table 2: Global Configuration

Chapter 3 Configuration Groups

3.1 ConsumerGroup

A **ConsumerGroup** contains two elements:

- A **DefaultConsumer** element, which you can use to specify a default **Consumer** component. If a default **Consumer** is not specified in the **ConsumerGroup**, EMA uses the first Consumer listed in the **ConsumerList**. For details on configuring a default **Consumer**, refer to Section 3.1.2.
- A **ConsumerList** element, which contains one or more **Consumer** components (each should be uniquely identified by a **<Name …/>** entry). The consumer component is the highest-level abstraction within an application and typically refers to **Channel** and/or **Dictionary** components which specify consumer capabilities.

For a generic **ConsumerGroup** XML schema, refer to Section 3.1.1.

For details on configuring a **ConsumerGroup**, refer to Section 3.1.3.

For a list of parameters you can use in configuring a **ConsumerGroup**, refer to Section 3.1.4.

3.1.1 Generic XML Schema for ConsumerGroup

The generic XML schema for **ConsumerGroup** is as follows:

```
<ConsumerGroup>
    <DefaultConsumer value="VALUE"/>
    <ConsumerList>
        <Consumer>
            <Name value="VALUE"/>
            ...
        </Consumer>
    </ConsumerList>
</ConsumerGroup>
```

3.1.2 Setting a Default Consumer

If a **DefaultConsumer** is not specified, then the EMA uses the first **Consumer** component in the **ConsumerGroup**. However, you can specify a default consumer by including the following parameter on a unique line inside **ConsumerGroup** but outside **ConsumerList** (for an example, refer to Appendix A).

```
<DefaultConsumer value="VALUE"/>
```

3.1.3 Configuring Consumers in a ConsumerGroup

To configure a **Consumer** component, add the appropriate parameters to the target consumer in the XML schema, each on a unique line (for a list of available **ConsumerGroup** parameters, refer to Section 3.1.4).

3.1.4 Consumer Entry Parameters

Use the following parameters when configuring a **ConsumerGroup** in EMA.

PARAMETER	TYPE	DEFAULT	DESCRIPTION
Channel	String	N/A	Specifies the channel that the **Consumer** component should use. This channel must match the **Name** parameter from the appropriate **<Channel>** entry in the **ChannelGroup** configuration. If **Channel** is not specified, the EMA resorts to default channel behavior when needed. For further details on the **<Channel>** entry and default behaviors, refer to Section 3.2.
Dictionary	String	N/A	Specifies how the consumer should access its dictionaries (it must match the **Name** parameter from the appropriate **<Dictionary>** entry in the **DictionaryGroup** configuration). If **Dictionary** is not specified, the EMA uses the channel's dictionary when needed. For further details on this default behavior, refer to Section 3.3.
DictionaryRequestTimeOut	long	45,000	Specifies the amount of time (in milliseconds) the application has to download dictionaries from a provider before the **OmmConsumer** throws an exception. If set to **0**, EMA will wait for a response indefinitely.
DirectoryRequestTimeOut	long	45,000	Specifies the amount of time (in milliseconds) the provider has to respond with a source directory refresh message before the **OmmConsumer** throws an exception. If set to **0**, EMA will wait for a response indefinitely.
DispatchTimeoutApiThread	int	0	Specifies the duration (in microseconds) for which the internal EMA thread is inactive before going active to check whether a message was received. If set to zero, the EMA internal thread goes active only if it gets notified about a received message.
ItemCountHint	long	100,000	Specifies the number of items the application expects to request. If set to **0**, EMA resets it to **1024**. For better performance, the application can set this to the approximate number of item requests it expects.
LoginRequestTimeOut	long	45,000	Specifies the amount of time (in milliseconds) the provider has to respond with a login refresh message before the **OmmConsumer** throws an exception. If set to **0**, EMA will wait for a response indefinitely.
MaxDispatchCountApiThread	long	100	Specifies the maximum number of messages the EMA dispatches before taking a real-time break.
MaxDispatchCountUserThread	long	100	Specifies the maximum number of messages the EMA can dispatch in a single call to the `OmmConsumer::dispatch()`.

Table 3: Consumer Group Parameters

PARAMETER	TYPE	DEFAULT	DESCRIPTION
MaxOutstandingPosts	long	100,000	Specifies the maximum allowable number of on-stream posts waiting for an acknowledgment before the **OmmConsumer** disconnects.
Name	String	N/A	Specifies the name of this **Consumer** component. **Name** is required when creating a **Consumer** component. You can use any value for **Name**.
ObeyOpenWindow	int	1	Specifies whether the **OmmConsumer** obeys the **OpenWindow** from services advertised in a provider's Source Directory response. Available values include: • 0 (false) • 1 (true)
PostAckTimeout	long	15,000	Specifies the length of time (in milliseconds) a stream waits to receive an ACK for an outstanding post before forwarding a negative acknowledgment to the application. If set to **0**, EMA will wait for a response indefinitely.
RequestTimeout	long	15,000	Specifies the amount of time (in milliseconds) the **OmmConsumer** waits for a response to a request before sending another request. If set to **0**, EMA will wait for a response indefinitely.
ServiceCountHint	long	513	Sets the size of directory structures for managing services. If the application specifies **0**, EMA resets it to **513**.

Table 3: Consumer Group Parameters (Continued)

3.2 Channel Group

The **ChannelGroup** contains a **ChannelList**, which contains one or more **Channel** entries (each uniquely identified by a **<Name …/>** entry). Each channel includes a set of connection parameters for a specific connection or connection type.

There is no default channel group. If a consumer application needs a specific channel group, you should specify this in the appropriate **Consumer** section (for details on configuring the **Consumer** component, refer to Section 3.1.3).

- For a generic **ChannelGroup** XML schema, refer to Section 3.2.1.
- For a list of universal parameters you can use in configuring any type of **Channel** regardless of the channel type, refer to Section 3.2.2.
- For a list of parameters you can use only when configuring a **Channel** whose channel type is **RSSL_SOCKET**, refer to Section 3.2.3.

3.2.1 Generic XML Schema for ChannelGroup

The top-level XML schema for the **ChannelGroup** is as follows:

```
<ChannelGroup>
    <ChannelList>
        <Channel>
            <Name value="VALUE"/>
            …
        </Channel>
    </ChannelList>
</ChannelGroup>
```

3.2.2 Universal Channel Entry Parameters

You can use the following parameters in any **<Channel>** entry, regardless of the **ChannelType**.

PARAMETER NAME	TYPE	DEFAULT	NOTES
ChannelType	String	RSSL_SOCKET	Specifies the type of channel or connection used to connect to the server. Calling the host function can change this field. For details on this event, refer to Section 4.3.2. Use enumeration values with EMA's programmatic configuration (for details, refer to in Section 4.4) Currently **RSSL_SOCKET (0)** is the only available value.
ConnectionPingTimeout	long	30000	Specifies the duration (in milliseconds) after which the EMA terminates the connection if it does not receive communication or pings from the server.
GuaranteedOutputBuffers	long	100	Specifies the number of guaranteed buffers (allocated at initialization time) available for use by each RsslChannel when writing data. Each buffer is created to contain **maxFragmentSize** bytes. For details on RsslChannel and **maxFragmentSize**, refer to the *Transport API Developers Guide*.
InterfaceName	String	""	Specifies a character representation of the IP address or hostname of the local network interface over which the EMA sends and receives content. **InterfaceName** is for use in systems that have multiple network interface cards. If unspecified, the default network interface is used.
MsgKeyInUpdates	int	1	Sets EMA to fill in message key values on updates using the message key provided with the request. Possible values are: • **0** (false): Do not fill in the message's key values (values received from the wire are preserved). • **1** (true): Fill in the message's key values (values received from the wire are overridden).
Name	String		Specifies the **Channel**'s name.
NumInputBuffers	long	10	Specifies the number of buffers used to read data. Buffers are sized according to **maxFragmentSize**. For details on RsslChannel and **maxFragmentSize**, refer to the *Transport API Developers Guide*.
ReconnectAttemptLimit	int	-1	Specifies the maximum number of times the **OmmConsumer** attempts to reconnect to a channel when it fails. If set to **-1**, the **OmmConsumer** continually attempts to reconnect.
ReconnectMaxDelay	int	5000	The maximum amount of time the **OmmConsumer** waits (in milliseconds) before attempting to reconnect a failed channel. Refer also to the preceding **ReconnectMinDelay** parameter.

Table 4: Universal <Channel> Parameters

PARAMETER NAME	TYPE	DEFAULT	NOTES
ReconnectMinDelay	int	1000	Specifies the minimum amount of time the **OmmConsumer** waits (in millseconds) before attempting to reconnect a failed channel. The time **OmmConsumer** waits between each connection attempt increases with each attempt, from **reconnectMinDelay** to **reconnectMaxDelay**.
SysRecvBufSize	long	0	Specifies the size (in KB) of the system's receive buffer for this channel.
SysSendBufSize	long	0	Specifies the size (in KB) of the system's send buffer for this channel.
XmlTraceToStdout	int	0	Specifies whether EMA traces its messages in XML format to stdout. Possible values are: • **0** (false): Turns off tracing. • **1** (true): Turns on tracing to stdout.

Table 4: Universal <Channel> Parameters (Continued)

3.2.3 Parameters for Use with Channel Type: RSSL_SOCKET

In addition to the universal parameters listed in Section 3.2.2, you can use the following parameters to configure a channel whose type is **RSSL_SOCKET**.

PARAMETER NAME	TYPE	DEFAULT	NOTES
CompressionThreshold	long	30	Sets the message size threshold (in bytes, the allowed value is 30-Integer.MAX_VALUE), above which all messages are compressed (thus individual messages might not be compressed). Different compression types have different behaviors and compression efficiency can vary depending on message size.
CompressionType	String	None	Specifies the EMA's preferred type of compression. Compression is negotiated between the client and server: if the server supports the preferred compression type, the server will compress data at that level. Available values are: • **None** (0) • **ZLib** (1) • **LZ4** (2) **Note:** A server can be configured to force a particular compression type, regardless of client settings.
Host	String	localhost	Specifies the host name of the server to which the EMA connects. The parameter value can be a remote host name or IP address.
Port	String	14002	Specifies the port on the remote server to which the EMA connects.
TcpNodelay	int	1	Specifies whether to use Nagle's algorithm when sending data. Available values are: • **0**: Send data using Nagle's algorithm. • **1**: Send data without delay.

Table 5: Parameters for Channel Type: RSSL_SOCKET

3.3 Dictionary Group

The **DictionaryGroup** contains a **DictionaryList**, which contains one or more **Dictionary** components (each uniquely identified by a **<Name …/>** entry). Each **Dictionary** component defines parameters relating to how the dictionary is accessed.

3.3.1 Generic XML Schema for DictionaryGroup

The top-level XML schema for **DictionaryGroup** is as follows:

```
<DictionaryGroup>
    <DictionaryList>
        <Dictionary>
            <Name value="…"/>
            …
        </Dictionary>
    </DictionaryList>
</DictionaryGroup>
```

3.3.2 Dictionary Entry Parameters

Use the following parameters when configuring a **Dictionary** entry in the EMA.

PARAMETER NAME	TYPE	DEFAULT	NOTES
DictionaryType	String	ChannelDictionary	Specifies the dictionary loading mode. Possible values are: • **FileDictionary** (0): The EMA loads the dictionaries from the files specified in the parameters **RdmFieldDictionaryFileName** and **EnumTypeDefFileName**. • **ChannelDictionary** (1): The EMA downloads dictionaries by requesting the dictionaries from the upstream provider.
EnumTypeDefFileName	String		Sets the location of the **EnumTypeDef** file.
Name	String		Sets a unique name for a Dictionary component in the **DictionaryList**.
RdmFieldDictionaryFileName	String		Sets the location of the **RdmFieldDictionary**.

Table 6: Dictionary Group Parameters

Chapter 4 EMA Configuration Processing

4.1 Default Configuration

The EMA configuration is determined by hard-coded behaviors, any customized beahviors specified in **EmaConfig.xml**, programmatic changes, and other internal processing. All of these items affect the configuration used by application components. This chapter discusses how the application configuration is derived.

Each application must eventually instantiate an ***OmmConsumer*** object. Constructors for OmmConsumer require a ***OmmConsumerConfig*** object. The ***OmmConsumerConfig*** constructor takes no arguments, but it does read and process an optional XML file (**EmaConfig.xml**), which applications can use to modify EMA's default behavior.

EMA provides a hard-coded configuration for use whenever an **OmmConsumerConfig** object is instantiated without an **EmaConfig.xml** file in the run-time environment. The resulting EMA configuration is created by taking the defaults from the various configuration groups. For example, the default (hard-coded) behavior for a **Channel** adheres to the following configuration:

- **ChannelType** value="RSSL_SOCKET"
- **CompressionType** value="None"
- **TcpNoDelay** value="1"
- **Host** value="localhost"
- **Port** value="14002"

Note that unlike EMA's default behavior of choosing the first **Consumer** component in the **ConsumerList**, EMA applications will not choose the first **Channel** or **Dictionary** in their respective lists. Instead, if an application wants to use a specific channel or dictionary configuration, the application must explicitly configure it in the appropriate **Consumer** section of the XML file.

For specifics on EMA's default configuration, refer to Section 2.2.

4.2 Processing EmaConfig.xml

Except for the parameter **DefaultConsumer**, all configuration elements defined in the**EmaConfig.xml** file must be wrapped within a component definition (i.e., **Consumer**, **Channel**, or **Dictionary**) or they will be ignored. This section includes some examples that illustrate this requirement. Appendix A illustrates the proper placement of **DefaultConsumer** within **EmaConfig.xml**.

4.2.1 Use of the Correct Order in the XML Schema

Consider the following snippet from an **EmaConfig.xml** (only those parts needed for the example are included). In this snippet, the application creates a consumer with a **Name** of **Consumer_1**.

```
<ConsumerGroup>
    <ConsumerList>
        <Consumer>
            <Name value="Consumer_1"/>
        </Consumer>
    </ConsumerList>
</ConsumerGroup>
```

4.2.2 Processing the Consumer “Name”

The EMA is hard-coded to use a default consumer of **EmaConsumer**. However, you can change this by using **EmaConfig.xml**. When you use the XML file, the default **Consumer Name** is either specified by the **DefaultConsumer** element, or if this parameter is not set, then the EMA application will default to the name of the first Consumer component.

- If **DefaultConsumer** uses an invalid name (i.e., no **Consumer** components in the XML file use that name), the EMA throws an exception indicating that **DefaultConsumer** is invalid.
- If the **EmaConfig.xml** has no **Consumer** components, the EMA application uses **EmaConsumer**.

4.3 Configuring EMA Using Function Calls

From an application standpoint, instantiating an OmmConsumerConfig object creates the initial configuration from the **EmaConfig.xml** file. Certain variables can then be altered via function calls on the OmmConsumerConfig object.

Note: Function calls override any settings in the **EmaConfig.xml** file.

4.3.1 EMA Function Calls

You can use the following function calls in an EMA application:

FUNCTION	DESCRIPTION
addAdminMsg(ReqMsg)	Populates part of or all of the login request message, directory request message, or dictionary request message according to the specification discussed in the *EMA Reuters Domain Models (RDM) Usage Guide* specific to the programming language you use.
applicationId(String)	Sets the **applicationId** variable. **applicationId** has no default value.
consumerName(String)	Sets the consumer name. If a consumer does not exist with that name, the application throws an exception.
host(String)	Sets the host and port parameters. For details, refer to Section 4.3.2.
operationModel(OperationModel)	Sets the operation model to either `OperationModel.API_DISPATCH` (which is the default) or `OperationModel.USER_DISPATCH`.
password(String)	Sets the **password** variable. **password** has no default value.
position(String)	Sets the **position** variable. **position** has no default value.
username(String)	Sets the **username** variable. If **username** is not set, the application extracts a username from the run-time environment.

Table 7: EMA Function Calls

4.3.2 Using the `host( )` Function: How "Host" and "Port" are Processed

Because the **Host** and **Port** parameters both have global default values (**localhost** and **14002** respectively), if an OmmConsumerConfig object exists, its **Host** and **Port** will always have values (either the default value or some value specified within **EmaConfig.xml**).

If you wish, you can have the application reset both host and port values by calling the **`host( String )`** method on the **`OmmConsumerConfig`** object using the syntax: ***HostValue*:*PortValue***.

Note: Calling the **`host()`** function results in the **channelType** (refer to Section 3.2.2) being set to **RSSL_SOCKET**, regardless of any previous setting for that configuration element.

Host and **Port** values observe the following rules when updating due to the **`host( String )`** method:

- If the host parameter is missing or empty, then host and port reset to their global default values (**localhost:14002**)
- If the host parameter is set to the string **":"**, then host and port reset to their global default values (**localhost:14002**)
- If the host parameter is a string (not containing a **:**), then host is set to that string and port resets to its default value (**14002**).
- If the parameter begins with a **:** and is followed by some text, then host is set to its global default value (**localhost**) and port is set to that text.
- If the parameter is ***HostValue*:*PortValue***, where both ***HostValue*** and ***PortValue*** have values, then host is set to ***HostValue*** and port is set to ***PortValue***.

Appendix A EmaConfig.xml Configuration File

This is the current version of the **EmaConfig.xml** file distributed with the training examples:

```
<?xml version="1.0" encoding="UTF-8"?>
<EmaConfig>

<ConsumerGroup>
    <!-- DefaultConsumer parameter defines which consumer configuration is used by OmmConsumer -->
    <!-- if application does not specify it through OmmConsumerConfig::consumerName() -->
    <!-- first consumer on the ConsumerList is a default consumer if this parameter is not specified -->
    <DefaultConsumer value="Consumer_1"/>
    <ConsumerList>
        <Consumer>
            <Name value="Consumer_1"/>

            <!-- Channel is optional: defaulted to "RSSL_SOCKET + localhost + 14002" -->
            <Channel value="Channel_1"/>

            <!-- Dictionary is optional: defaulted to "ChannelDictionary" -->
            <Dictionary value="Dictionary_1"/>
        </Consumer>
        <Consumer>
            <Name value="Consumer_2"/>
            <Channel value="Channel_2"/>
            <Dictionary value="Dictionary_2"/>
        </Consumer>
    </ConsumerList>
</ConsumerGroup>
<ChannelGroup>
    <ChannelList>
        <Channel>
            <Name value="Channel_1"/>

            <!-- ChannelType possible values are: -->
            <!-- ChannelType::RSSL_SOCKET    - TCP IP connection type -->
            <ChannelType value="ChannelType::RSSL_SOCKET"/>

            <!-- CompressionType is optional: defaulted to None -->
            <!-- possible values: None, ZLib, LZ4 -->
            <CompressionType value="CompressionType::None"/>
            <GuaranteedOutputBuffers value="5000"/>

            <!-- ConnectionPingTimeout is optional: defaulted to 30000 -->
            <ConnectionPingTimeout value="30000"/>
```

```
            <!-- TcpNodelay is optional: defaulted to 1 -->
            <!-- possible values: 1 (tcp_nodelay option set), 0 (tcp_nodelay not set) -->
            <TcpNodelay value="1"/>
            <Host value="localhost"/>
            <Port value="14002"/>
        </Channel>
        <Channel>
            <Name value="Channel_2"/>
            <ChannelType value="ChannelType::RSSL_SOCKET"/>
            <CompressionType value="CompressionType::None"/>
            <GuaranteedOutputBuffers value="5000"/>
            <Host value="122.1.1.100"/>
            <Port value="14002"/>
        </Channel>
    </ChannelList>
</ChannelGroup>
<DictionaryGroup>
    <DictionaryList>
        <Dictionary>
            <Name value="Dictionary_1"/>
            <!-- DictionaryType is optional: defaulted to ChannelDictionary" -->
            <!-- possible values: FileDictionary, ChannelDictionary -->
            <!-- if DictionaryType is set to ChannelDictionary, file names are ignored -->
            <DictionaryType value="DictionaryType::ChannelDictionary"/>
        </Dictionary>
        <Dictionary>
            <Name value="Dictionary_2"/>
            <DictionaryType value="DictionaryType::FileDictionary"/>

            <!-- dictionary names are optional: defaulted to RDMFieldDictionary and enumtype.def -->
            <RdmFieldDictionaryFileName value="./RDMFieldDictionary"/>
            <EnumTypeDefFileName value="./enumtype.def"/>
        </Dictionary>
    </DictionaryList>
</DictionaryGroup>
</EmaConfig>
```

Document ID: EMAJ300CG.160
Date of issue: 29 April 2016