ELEKTRON MESSAGE API V3.0

CONFIGURATION GUIDE

C++ EDITION

Document ID: EMA 001.001
Document Version: 1.0
Date of Issue: 08 May 2015

THOMSON REUTERS

Contents

Tables

Chapter 1 Introduction

1.1 Introduction

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

Configuration works in a similar fashion across all platforms.

1.2 Audience

Software developers and local site administrators will use this document to understand configuration parameters.

Software developers can get further information from the *EMA Developer's Guide*.

1.3 Using This Document

The material presented in this guide is divided into the following sections:

CHAPTER	DESCRIPTION
Chapter 1	About this Manual
Chapter 2	EMA Configuration Parameters
Appendix A	Configuration Files

Table 1: Chapter Overview

1.4 Definitions

- ***Group***: Related set of configuration parameters for a specific EMA component (e.g., ChannelGroup)
- ***List***: A list of components belonging to a group (e.g., ChannelList)
- ***Component***: A specific component (e.g., Channel). Since lists can have multiple components, each component must have a 'name' field for identification purposes.
- ***Field***: actual configurable parameter
- ***Default Value***: if specified, the value used by the API if not specified by the user. In general, items with default values are required by the API.
- ***Allowed value***: Specific values or a range of values allowed for a field.

1.5 References

[1] *EMA …*

[2] *Another EMA ...*

[3] *Reuters Domain Models (RDM) Usage Guide, C++ Edition*

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.
- Mark up the PDF using the Comment feature in Adobe Reader. After adding your comments, you can submit the entire PDF to Thomson Reuters by clicking **Send File** in the **File** menu. Use the apidocumentation@thomsonreuters.com address.

1.7 Conventions

- Parameters, filenames, and directories appear in **bold** text.
- Variable values appear in *italics*
- Newly introduced EMA terms appear in ***Bold Italics***..
- Data Type usage within the configuration repository is described in the following table:

CONFIGURATION DATA TYPE	FLAT FILE
EmaString	String
Enumeration	Specific text, as indicated in the field description
Int64	Signed long integer
UInt64	Unsigned long integer

Table 2: Configuration Data Types

- Within XML files, the value for an individual field is specified as **<fieldName value="value for field"/>**. The *fieldName* cannot contain white space. The *value* is always included within double quotes; those quotes will be omitted from the field descriptions in the remainder of this document.

Even though enumerations have text values (i.e., RSSL_CONN_TYPE_SOCKET), within the software, those text values are represented as numbers. When enumerations are introduced, those numbers will be listed along with the text values because the numbers are required for programmatic configuration.

Chapter 2 EMA Configuration Parameters

The EMA configuration is written using a simple XML schema, some of which can be changed via software function calls. The initial configuration, which is compiled into the EMA software, defines a minimal set of configuration parameters. EMA users can also supply an XML file, named **EmaConfig.xml,** which can be used to specify configuration parameters. In addition, parameters can be changed via software function calls or via programmatic interfaces.

The EMA configuration data is divided into four groups:

- Consumer – highest level description of application. Typically used to select entries from channel, logger, and dictionary groups.
- Channel – describes various connection alternatives and provides configuration alternatives for those connections.
- Logger – logging alternatives and associated configuration parameters.
- Dictionary – location information for dictionary alternatives.

The remainder of this section discusses the four configuration groups and the configuration parameters available to each group.

2.1 EMA Global Configuration Parameters

Some EMA configuration parameters are global parameters to the configuration, even when they actually affect only one of the configuration groups described in the next section.

Default values exist for all parameters used within EMA. The defaults are used when required. For example, here are a few parameters and their default values:

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
Hostname	EmaString		localhost	
Port	EmaString		14002	
DefaultConsumer	EmaString		EmaConsumer	ignored unless no Consumer components are configured
LoggerSeverity	Enumeration	Verbose (0) Success (1) Warning (2) Error (3) NoLogMsg (4)	Success	see Section 2.2.3.1
LoggerType	Enumeration	File (0) Stdout (1)	File	
FileName	EmaString		emaLog	
RdmFieldDictionaryFileName	EmaString		./RDMFieldDictionary	
EnumTypeDefFileName	EmaString		./enumtype.def	

Table 3 Global Configuration

When the EMA library needs a parameter, it uses the default value unless the user had correctly specified an alternate value, either via the **EmaConfig.xml** file, function calls, or programmatic methods.

2.2 EMA Configuration Groups

2.2.1 ConsumerGroup

The XML schema for the ConsumerGroup is written as follows:

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

The **ConsumerGroup** contains a **ConsumerList**, which contains one or more **Consumer** components, with each of those components having a <*Name* …/> entry which specifies the name of the consumer.

The consumer is the highest-level abstraction within an application and typically refers to **Channel**, **Logger**, and/or **Dictionary** components to specify consumer capabilities. For example, if the configuration included one or more logger schemas, a **Consumer** configuration might include

```
<Logger value="…"/>
```

to pick a specific logger.

Users may also specify a default consumer by including

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

inside the ConsumerGroup but outside the ConsumerList. If no *DefaultConsumer* is specified, then the first **Consumer** component in the **ConsumerGroup** is used.

2.2.1.1 ConsumerGroup Parameters

The following tables lists parameters that the EMA software is prepared to process and any constraints on those parameters.

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
Name	EmaString		n/a	
Channel	EmaString	one of the names associated with an entry in the **ChannelGroup** configuration.	even though no default exists for *Channel*, the software will use appropriate defaults when a channel configuration is needed. These defaults are discussed in	

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
			Section 2.2.2	
Logger	EmaString	one of the names associated with an entry in the LoggerGroup configuration	even though no default exists for *Logger*, the software will use appropriate defaults when a logger configuration is needed. These defaults are discussed in 2.2.3	
Dictionary	EmaString	one of the names associated with an entry in the DictionaryGroup configuration	even though no default exists for *Dictionary*, the software will use appropriate defaults when a dictionary configuration is needed. These defaults are discussed in 2.2.4	
DirectoryRequestTimeOut	UInt64		45 seconds	If the application is downloading directories from the provider but this does not occur in the time specified, the application will throw an exception
LoginRequestTimeOut	UInt64		45 seconds	if the provider cannot be reached in the time specified, the application will throw an exception.
ItemCountHint	UInt64		100,000	if configured to 0, will be reset to 513
ServiceCountHint	UInt64		513	if configured to 0, will be reset to 513
ObeyOpenWindow	UInt64	0 (value = 0), 1 (value > 0)	1	
PostAckTimeout	UInt64		15,000	
RequestTimeout	UInt64		15,000	
MaxOutstandingPosts	UInt64		100,000	
DispatchTimeoutApiThread	Int64		-1	if less than zero, application will not timeout when submitting a request
CatchUnhandledException	UInt64	0 (value = 0), 1 (value > 0)		
MaxDispatchCountApiThread	UInt64		100	
MaxDispatchCountUserThread	UInt64		1	
PipePort	Int64		9001	

Table 4 Consumer Group Elements

2.2.2 ChannelGroup

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

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

The ChannelGroup contains a ChannelList, which contains one or more Channel entries, with each of those entries having a <Name …/> entry which specifies the name of the channel.

The channel describes connection parameters for a specific connection or connection type.

There is no default channel group. If the application needs a specific channel group, the specification should occur in the appropriate **Consumer** specification.

2.2.2.1 ChannelGroup Parameters

The following tables lists parameters that the EMA software is prepared to process and any constraints on those parameters.

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
Name	EmaString			
ChannelType	Enumeration	RSSL_ SOCKET (0) RSSL_ ENCRYPTED (1) RSSL_HTTP (2) RSSL_RELIABLE_MCAST (3)	RSSL_SOCKET	see 2.3.2.2 for a note regarding potential changes to this field as a result of calling the **host** function.
CompressionType	Enumeration	None (0) ZLib (1) LZ4 (2)	None	
GuaranteedOutputBuffers	UInt64		100	
ConnectionPingTimeout	UInt64		30000	
TcpNodelay	UInt64	0 (value = 0), 1 (otherwise)	1	if 0, then tcp_nodelay not set
Host	EmaString		localhost	
Port	EmaString		14002	

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
XmlTraceFileName	EmaString		EmaTrace	file name for file containing XML trace output if tracing is selected
XmlTraceToFile	UInt64	0 (value = 0), 1 (otherwise)	0	tracing off if value is 0
XmlTraceMaxFileSize	Int64		100000000	
XmlTraceToStdout	UInt64	0 (value = 0), 1 (otherwise)	0	
XmlTraceToMultipleFiles	UInt64	0 (value = 0), 1 (otherwise)	0	
XmlTraceWrite	UInt64	0 (value = 0), 1 (otherwise)	1	
XmlTraceRead	UInt64	0 (value = 0), 1 (otherwise)	1	
InterfaceName	EmaString		“”	
ReconnectAttemptLimit	Int64		-1	
ReconnectMinDelay	Int64		1000	
ReconnectMaxDelay	Int64		5000	

Table 5 Channel Group Elements

2.2.3 Logger Group

The top-level XML schema for the LoggerGroup is written as follows:

```
<LoggerGroup>
 <LoggerList>
   <Logger>
     <Name value="…"/>
     …
   </Logger>
 </LoggerList>
</LoggerGroup>
```

The **LoggerGroup** contains a **LoggerList**, which contains one or more **Logger** components, with each of those entries having a <Name …/> entry which specifies the name of the **Logger** component.

The **Logger** component describes parameters for a logging facility.

2.2.3.1 LoggerGroup Parameters

The following tables lists parameters that the EMA software is prepared to process and any constraints on those parameters.

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
Name	EmaString			
LoggerType	Enumeration	File (0) Stdout (1)	File	
LoggerSeverity	Enumeration	Verbose (0) Success (1) Warning (2) Error (3) NoLogMsg (4)	Success	Each severity level includes all messages for that level plus all messages for higher levels. For examples, level “Verbose” will result in all possible logger messages, while level “Error” will exclude messages with levels “Verbose,” “Success,” and “Warning.”
FileName	EmaString		“emaLog_<pid>.log”	Meaningful only if LoggerType is File
IncludeDateInLoggerOutput	UInt64	0 1	0	If set to 0, then log messages include the time only; if set to 1, then both date and time are printed.

Table 6 Logger Group Elements

2.2.4 Dictionary Group

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

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

The **DictionaryGroup** contains a **DictionaryList**, which contains one or more **Dictionary** entries, with each of those entries having a <Name …/> entry which specifies the name of the **Dictionary** component.

The **Dictionary** component describes parameters for accessing necessary dictionaries.

2.2.4.1 DictionaryGroup Parameters

The following tables lists parameters that the EMA software is prepared to process and any constraints on those parameters.

PARAMETER NAME	TYPE	ALLOWED VALUES	DEFAULT	NOTES
Name	EmaString		n/a	
DictionaryType	Enumeration	FileDictionary (0) ChannelDictionary (1)	ChannelDictionary	
RdmFieldDictionaryFileName	EmaString			location of RdmFieldDictionary
EnumTypeDefFileName	EmaString			location of EnumTypeDef file

Table 7 Dictionary Group Elements

2.3 EMA Configuration Processing

The EMA configuration is affected by precompiled configuration data, possibly a user-provided configuration file, programmatic changes, and other internal processing. All of these items feed into a configuration used by application components. This section discusses how the application configuration is derived.

Each application must eventually instantiate an *OmmConsumer* object. Both constructors for the *OmmConsumer* require a *OmmConsumerConfig* object. The *OmmConsumerConfig* constructor takes no arguments, but it does read and process an optional file called **EmaConfig.xml**, which applications can use to modify the minimal configuration built into the software.

The base configuration data is compiled into the EMA library. These variables are global to the group in which they are used. For example, the field *FileName* is used within the **LoggerGroup**, so if a **LoggerGroup** configuration does not explicitly specify a value for *FileName*, then the *FileName* default value defined within the software will be used. These defaults are listed in the description of the configuration groups previous in this document.

2.3.1 Default configuration

EMA supports a default configuration, which is the configuration that results if an *OmmConsumerConfig* object is instantiated and there is no **EmaConfig.xml** file present in the run-time environment. The configuration that results in this situation is the one created by taking the defaults from the various configuration groups. For example, the default configuration for Channel include **ChannelType** *RSSL_Socket*, no compression, tcp_nodelay set, **host** *localhost*, **port** *14002*, and no xml tracing. Other defaults for the channel are found in Channel Group Elements.

Note that unlike choosing a **Consumer** component, EMA applications will not choose the first *Logger* component in a *LoggerList* or the first Channel or the first Dictionary; instead if an application wants to use a specific channel, logger, or dictionary configuration, that select must be made in the configuration for the applicable **Consumer**.

2.3.2 EmaConfig.xml configuration processing

Note that, with one exception, all of the configuration elements defined in the **EmaConfig.xml** file must be contained within component definitions or they will be ignored. Here are some examples to illustrate this requirement.

Assume the following configuration is defined within **EmaConfig.xml** (note: parts of the **EmaConfig.xml** not needed for this discussion are omitted):

```
<ConsumerGroup>
 <ConsumerList>
   <Consumer>
     <Name value="Consumer_1"/>
     <Logger value="Logger_2"/>
   </Consumer>
 </ConsumerList>
</ConsumerGroup>
<LoggerGroup>
 <LoggerList>
   <Logger>
     <Name value="Logger_2"/>
     <LoggerType value="LoggerType::File"/>
```

```
    <FileName value="emaLogfile"/>
   </Logger>
 </LoggerList>
</LoggerGroup>
```

In this case, the application will create a consumer with *Name* **Consumer_1** and this consumer will log to a file named **emaLogfile**.

Now assume that the line

```
      <FileName value="emaLogfile"/>
```

was not present in the configuration. In this case, the application would use the default *FileName* **emaLog** as indicated in the *FileName* description found in Logger Group Elements.

One might think that either of the following configuration snippets make sense:

```
<LoggerGroup>
 <FileName value="…"/>
 <LoggerList>
```

or

```
<LoggerGroup>
   <LoggerList>
   <FileName value="..."/-->
```

However, in both these cases the *FileName* has no effect because the configuration implementation does not allow the default value for a configuration element to be changed via the **EmaConfig.xml** file.

The one exception to the rule that configuration parameters must be defined within a component configuration is the **DefaultConsumer** element. This may be defined as follows:

```
<ConsumerGroup>
 <DefaultConsumer value="Consumer_1"/>
 <ConsumerList>
```

2.3.2.1 Consumer Name

The default **ConsumerName** is *EmaConsumer*.

If **EmaConfig.xml** exists, then the default **ConsumerName** is the value specified by the **DefaultConsumer** element (see above). If **DefaultConsumer** is not set, then the name of the first Consumer component is used.

If the **DefaultConsumer** element has a value for a invalid name (i.e., no Consumer components have that name), the **DefaultConsumer** will be ignored.

If no Consumer components have been defined, the default **ConsumerName** is used.

Once the **OmmConsumerConfig** object exists, one can call the **consumerName** method on that object, with a single parameter being the new **ConsumerName**. If no consumer has been defined with that name, the application will throw an exception.

2.3.2.2 Host and Port Names

Slightly special handling exists for host and port names. As with other configuration variables, these have default values (*localhost* and *14002*, respectively) that are used if no **EmaConfig.xml** file exists or if a channel group is specified in the **EmaConfig.xml** file but without specifications for the host and/or port within the channel group.

Once the **OmmConsumerConfig** object exists, the host and port each have some value, either the default value or the value specified within **EmaConfig.xml**. The application may then call the **host(EmaString &)** method on the **OmmConsumerConfig** object. The **host(EmaString &)** parameter and the resulting values for the host and port are:

- if no parameter is specified, both host and port are set to their default values
- if the parameter is the empty string or the string ":",both host and port are set to their default values
- if the parameter is a string not containing a ':' character, the host is set to that string and the port is set to its default value.
- If the parameter begins with a ':' and is followed by some text, the hostname is set to the default value and the port is set to that text
- If the parameter is '*h:p*' where both *h* and *p* are non-empty, then the host is set to *h* and the port is set to *p*.

Note that calling the host() function results in the channelType (see 2.2.2.1) being set to RSSL_CONN_TYPE_SOCKET, regardless of any previous setting for that configuration element.

2.3.3 Function calls that directly change the configuration

From an application standpoint, the creation of the initial configuration from the DefaultXML.h and EmaConfig.xml files occurs when the application instantiates an OmmConsumerConfig object.

Certain variables can then be altered via function calls on the OmmConsumerConfig object. The following function calls are available:

- **username**(const EMAstring &) – the *username* variable has no default, but applications can set a username by calling this method. If not set, the application will extract a user name from the run-time environment.
- **password**(const EMAstring &) – the *password* variable has no default. This method can be used to set a password.
- **position**(const EMAstring &) – the *position* variable has no default, but applications can specify *position* using this method.
- **applicationId**(const EMAstring &) - the *applicationId* variable has no default, but applications can specify *applicationId* using this method.
- **host**(const EMAstring &) – see 2.3.2.2
- **operationModel**(OperationModel) – this function may be used to set the operation model to either of *OmmConsumerConfig::ApiDispatchEnum* or *OmmConsumerConfig::UserDispatchEnum*; the default is *OmmConsumerConfig::ApiDispatchEnum*.
- **consumerName** (EmaString &) see 2.3.2.1.

2.3.4 addAdminMsg Function

The addAdminMsg function is used to populate any or all of the login request message, directory request message, or dictionary request message according to the specification discussed in the *Reuters Domain Models (RDM) Usage Guide, C++ Edition*.

2.3.5 Programmatic Configuration

One more method exists to change the configuration. Applications can create a Map and then call the **config** method on an OmmConsumerConfig object, passing the Map as a parameter to the **config** method. This can be done multiple times,

if desired and each of the programmatic configurations will be applied to create the application's active configuration during the *OmmConsumer* instantiation.

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"/>

                        <!-- Logger is optional: defaulted to "File + Success" -->
                        <Logger value="Logger_1"/>

                        <!-- Dictionary is optional: defaulted to "ChannelDictionary" -->
                        <Dictionary value="Dictionary_1"/>
                </Consumer>
                <Consumer>
                        <Name value="Consumer_2"/>
                        <Channel value="Channel_2"/>
                        <Logger value="Logger_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::RSSL_HTTP      - Http tunnel connection type -->
                        <!-- ChannelType::RSSL_ENCRYPTED - Https tunnel connection type -->
                        <!-- ChannelType::RSSL_RELIABLE_MCAST - MCAST connection type (not supported yet) -->
                        <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>
                <Channel>
                        <Name value="Channel_3"/>
                        <ChannelType value="ChannelType::RSSL_ENCRYPTED"/>
                        <CompressionType value="CompressionType::None"/>
                        <GuaranteedOutputBuffers value="5000"/>
                        <Host value="122.1.1.100"/>
                        <Port value="14002"/>
                </Channel>
                <Channel>
                        <Name value="Channel_4"/>
                        <ChannelType value="ChannelType::RSSL_HTTP"/>
                        <CompressionType value="CompressionType::None"/>
                        <GuaranteedOutputBuffers value="5000"/>
                        <Host value="122.1.1.100"/>
                        <Port value="14002"/>
                </Channel>
        </ChannelList>
</ChannelGroup>
<LoggerGroup>
        <LoggerList>
                <Logger>
                        <Name value="Logger_1"/>

                        <!-- LoggerType is optional:  defaulted to "File" -->
                        <!-- possible values: Stdout, File -->
                        <LoggerType value="LoggerType::Stdout"/>

                        <!-- LoggerSeverity is optional: defaulted to "Success" -->
                        <!-- possible values: Verbose, Success, Warning, Error, NoLogMsg -->
                        <LoggerSeverity value="LoggerSeverity::Success"/>
                </Logger>
                <Logger>
                        <Name value="Logger_2"/>
                        <LoggerType value="LoggerType::File"/>
                        <!-- FileName is optional: defaulted to "emaLog_<process id>.log" -->
                        <FileName value="emaLog"/>
                        <LoggerSeverity value="LoggerSeverity::Success"/>
                </Logger>
        </LoggerList>
</LoggerGroup>
<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>
```