# Elektron Message API C++ Edition V3.0

## ELEKTRON MESSAGE API CONFIGURATION GUIDE

Document Version: 3.0
Date of issue: 30 June 2015
Document ID: EMA300CG.150

THOMSON REUTERS

# Contents

# Chapter 1 Guide 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. 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 any specific component 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 (EMA). EMA 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 *EMA Developer's Guide*.

## 1.3 About EMA Configuration

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

EMA 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, logger, and dictionary groups.
- ***Channel***: Channel configuration data describe various connection alternatives and provides configuration alternatives for those connections.
- ***Logger***: Logger configuration data specify logging alternatives and associated parameters.
- ***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 Definitions

| DEFINITION | DESCRIPTION |
|---|---|
| Group | A 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). Because lists can have multiple components, each component must have a 'name' field for identification purposes. |
| Field | A configurable parameter. |
| Default Value | A default value is the value the API uses if a value is 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 that the field allows. |

**Table 1: Definitions**

## 1.5 Acronyms and Abbreviations

| ACRONYM | MEANING |
|---|---|
| ADH | Advanced Data Hub |
| ADS | Advanced Distribution Server |
| API | Application Programming Interface |
| ASCII | American Standard Code for Information Interchange |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol (Secure) |
| OMM | Open Message Model |
| QoS | Quality of Service |
| RDF | Reuters Data Feeds |
| 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 Modelling Language |
| UTF-8 | 8-bit Unicode Transformation Format |

**Table 2: Acronyms and Abbreviations**

## 1.6 References

1. *API Concepts Guide*
2. *Elektron Message API Developers Guide*
3. *Transport API C Edition Developers Guide*

## 1.7 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.

## 1.8 Document Conventions

This document uses the following types of conventions:

- Typographic
- Data Types
- Field and Text Values
- Diagrams

### 1.8.1 Typographic

- C structures, 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 XML examples (one or more lines of code) are shown in Courier New font against an orange background. For example:

```
/* decode contents into the filter list structure */
if ((retVal = rsslDecodeFilterList(&decIter, &filterList)) >= RSSL_RET_SUCCESS)
{
   /* create single filter entry and reuse while decoding each entry */
   RsslFilterEntry filterEntry = RSSL_INIT_FILTER_ENTRY;
```

### 1.8.2 Data Types

Data types within the configuration repository are as follows:

| DATA TYPE | DEFINITION |
|---|---|
| EmaString | String |
| Enumeration | Specific text, as indicated in the field description |
| Int64 | Signed long integer |
| UInt64 | Unsigned long integer |

**Table 3: Data Type Conventions**

### 1.8.3 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., RSSL_CONN_TYPE_SOCKET), in the software, text values are represented as numbers (which are required for programmatic configuration). When enumerations are introduced, the numbers are listed along with the text 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 example:

- EMA's globally default behavior is to log its messages at a **LoggerSeverity** level of **Success** to a file named **emaLog_*pid*.log** (where ***pid*** is the process ID). You can manually change the **LoggerSeverity** and the log filename by using **EmaConfig.xml**.
- By default (globally), the EMA does not XML trace to file (equivalent to **XmlTraceToFile value="0"**). You need to add this parameter only if you want to turn on XML tracing. If you turn on XML tracing (a non-default behavior), the EMA will trace to a file named **EmaTrace** (equivalent to **XmlTraceFileName value="EmaTrace"**).

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 |
|---|---|---|---|
| Hostname | EmaString | 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 | EmaString | 14002 | Specifies the port number on the server to which the EMA connects. |
| DefaultConsumer | EmaString | EmaConsumer | If consumer components are configured, this parameter is ignored. |
| LoggerSeverity | Enumeration | Success | Sets the level at which the EMA logs events.<br>For details, refer to Section 3.3.2. |
| LoggerType | Enumeration | File | Specifies the destination for output messages. The parameter value can be either `File` or `Stdout`. |
| FileName | EmaString | "emaLog_*pid*.log" | Specifies the name of log file (used when `LoggerType value= "File"`) |

**Table 4: Global Configuration**

| PARAMETER | TYPE | DEFAULT BEHAVIOR | NOTES |
|---|---|---|---|
| RdmFieldDictionaryFileName | EmaString | ./RDMFieldDictionary | Specifies the path and name of the **RdmFieldDictionary** file. |
| EnumTypeDefFileName | EmaString | ./enumtype.def | Specifies the path and name of the **enumtypeDef** dictionary file. |

**Table 4: Global Configuration (Continued)**

# 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**, **Logger**, 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).

For example, if your configuration includes logger schemas, you specify the desired logger schema by adding the following parameter inside the appropriate **Consumer** section:

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

Consumer components can use different logger schemas if the configuration includes more than one.

### 3.1.4 Consumer Entry Parameters

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

| PARAMETER | TYPE | ALLOWED VALUES | DEFAULT | DESCRIPTION |
|---|---|---|---|---|
| Name | EmaString | Any value (contained in quotes). | N/A | Specifies the name of this **Consumer** component. **Name** is required when creating a **Consumer** component. |
| Channel | EmaString | One of the names associated with an entry in the **ChannelGroup** configuration. | N/A | Specifies the channel that the **Consumer** component should use (it must match the **Name** parameter from the appropriate **<Channel>** entry).<br>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 |
| Logger | EmaString | One of the names associated with an entry in the **LoggerGroup** configuration. | N/A | Specifies a set of logging behavior the **Consumer** should exhibit (it must match the **Name** parameter from the appropriate **<Logger>** entry).<br>If **Logger** is not specified, the EMA uses a set of logger default behaviors. For further details on the **<Logger>** entry and default settings, refer to Section 3.3. |

**Table 5: Consumer Group Parameters**

| PARAMETER | TYPE | ALLOWED VALUES | DEFAULT | DESCRIPTION |
|---|---|---|---|---|
| Dictionary | EmaString | One of the names associated with an entry in the **DictionaryGroup** configuration | N/A | Specifies how the consumer should access its dictionaries (it must match the **Name** parameter from the appropriate **<Dictionary>** entry).<br>If **Dictionary** is not specified, the EMA uses the channel's dictionary when needed. For further details on this default behavior, refer to Section 3.4. |
| DictionaryRequestTimeOut | UInt64 | | 45,000 | Specifies the amount of time (in milliseconds) the application has to download dictionaries from a provider before the **OmmConsumer** throws an exception. |
| DirectoryRequestTimeOut | UInt64 | | 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. |
| LoginRequestTimeOut | UInt64 | | 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. |
| ItemCountHint | UInt64 | | 100,000 | If set to **0**, the EMA resets it to **513**. |
| ServiceCountHint | UInt64 | | 513 | If set to **0**, the EMA resets it to **513**. |
| ObeyOpenWindow | UInt64 | • 0 (false)<br>• 1 (true) | 1 | Specifies whether the **OmmConsumer** obeys the **OpenWindow** from services advertised in a provider's Source Directory response. |
| PostAckTimeout | UInt64 | | 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. |

**Table 5: Consumer Group Parameters (Continued)**

| PARAMETER | TYPE | ALLOWED VALUES | DEFAULT | DESCRIPTION |
|---|---|---|---|---|
| RequestTimeout | UInt64 | | 15,000 | Specifies the amount of time (in milliseconds) the **OmmConsumer** waits for a response to a request before sending another request. |
| MaxOutstandingPosts | UInt64 | | 100,000 | Specifies the maximum allowable number of on-stream posts waiting for an acknowledgment before the **OmmConsumer** disconnects. |
| DispatchTimeoutApiThread | Int64 | | -1 | Specifies the duration (in microseconds) for which the internal EMA thread is inactive before going active.<br>If set to less than zero, the EMA internal thread goes active only if it gets notified about a received message. |
| CatchUnhandledException | UInt64 | • 0 (false)<br>• 1 (true) | 1 | |
| MaxDispatchCountApiThread | UInt64 | | 100 | Specifies the maximum number of messages the EMA dispatches before taking a real-time break. |
| MaxDispatchCountUserThread | UInt64 | | 100 | Specifies the maximum number of messages the EMA can dispatch in a single call to the `OmmConsumer::dispatch()`. |
| PipePort | Int64 | | 9001 | Specifies the internal communication port. You might need to adjusted this port if it conflicts with other processes on the machine. |

**Table 5: 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 **ConsumerGroup** XML schema, refer to Section 3.2.1.
- For a list of parameters you can use in configuring a **ConsumerGroup**, refer to Section 3.2.2.

### 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 Channel Entry Parameters

Each **<Channel>** entry can have the following parameters.

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| Name | EmaString | | | Specifies the **Channel**'s name. |
| ChannelType | Enumeration | • RSSL_ SOCKET (0)<br>• RSSL_ ENCRYPTED (1)<br>• RSSL_HTTP (2)<br>• RSSL_RELIABLE_MCAST (3) | RSSL_SOCKET | Calling the host function can change this field. For details on this event, refer to Section 4.3.2. |

**Table 6: Channel Group Parameters**

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| CompressionType | Enumeration | • None (0)<br>• ZLib (1)<br>• LZ4 (2) | 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.<br>**Note:** If configured to do so, the server can force compression, regardless of client settings. |
| GuaranteedOutputBuffers | UInt64 | | 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.<br>For details on RsslChannel and **maxFragmentSize**, refer to the *Transport API Developers Guide*. |
| ConnectionPingTimeout | UInt64 | | 30000 | |
| TcpNodelay | UInt64 | • 0<br>• 1 | 1 | Specifies whether to use nagle's algorithm when sending data:<br>• **0**: The EMA sends data using Nagle's algorithm.<br>• **1**: The EMA sends data without delay. |
| Host | EmaString | | 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 | EmaString | | 14002 | Specifies the port on the remote server to which the EMA connects. |
| XmlTraceFileName | EmaString | | EmaTrace | Sets the name of the file to which to write XML trace output if tracing is selected. |

**Table 6: Channel Group Parameters (Continued)**

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| XmlTraceToFile | UInt64 | • 0 (false)<br>• 1 (true) | 0 | Using a value of 0 turns off tracing. |
| XmlTraceMaxFileSize | UInt64 | | 100000000 | |
| XmlTraceToStdout | UInt64 | • 0 (false)<br>• 1 (true) | 0 | |
| XmlTraceToMultipleFiles | UInt64 | • 0 (false)<br>• 1 (true) | 0 | |
| XmlTraceWrite | UInt64 | • 0 (false)<br>• 1 (true) | 1 | |
| XmlTraceRead | UInt64 | • 0 (false)<br>• 1 (true) | 1 | |
| InterfaceName | EmaString | | “” | Specifies a character representation of the IP address or hostname of the local network interface over which the EMA sends and receives content.<br>**InterfaceName** is for use in systems that have multiple network interface cards. If unspecified, the default network interface is used. |
| ReconnectAttemptLimit | Int64 | | -1 | Specifies the maximum number of times the **OmmConsumer** attempts to reconnect to a channel when it fails.<br>If set to **-1**, the **OmmConsumer** continually attempts to reconnect. |

**Table 6: Channel Group Parameters (Continued)**

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| ReconnectMinDelay | Int64 | | 1000 | Specifies the minimum amount of time the **OmmConsumer** waits (in millseconds) before attempting to reconnect a failed channel.<br>The time **OmmConsumer** waits between each connection attempt increases with each attempt, from **reconnectMinDelay** to **reconnectMaxDelay**. |
| ReconnectMaxDelay | Int64 | | 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 6: Channel Group Parameters (Continued)**

## 3.3 Logger Group

**LoggerGroup** contains a **LoggerList**, which contains one or more **Logger** components (each uniquely identified by a **<Name …/>** entry). A Logger component defines the parameters and behaviors for a single logging utility.

### 3.3.1 Generic XML Schema for LoggerGroup

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

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

### 3.3.2 Logger Entry Parameters

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

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| Name | EmaString | | | |
| LoggerType | Enumeration | • File (0)<br>• Stdout (1) | File | |
| LoggerSeverity | Enumeration | • Verbose (0)<br>• Success (1)<br>• Warning (2)<br>• Error (3)<br>• NoLogMsg (4) | Success | Severity levels aggregate messages so that a severity level includes all messages from higher levels (e.g., a setting of **1** would include any messages normally printed at levels **2** and **3**). |
| FileName | EmaString | | "emaLog_*pid*.log" | The EMA ignores this parameter if **LoggerType** is set to **Stdout** (1). |
| IncludeDateInLoggerOutput | UInt64 | • 0 (false)<br>• 1 (true) | 0 | • If set to **0**, log messages include the time only.<br>• If set to **1**, log messages include both date and time. |

**Table 7: Logger Group Parameters**

## 3.4 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.

### 3.4.1 Generic XML Schema for DictionaryGroup

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

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

### 3.4.2 Dictionary Entry Parameters

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

| PARAMETER NAME | TYPE | ALLOWED VALUES | DEFAULT | NOTES |
|---|---|---|---|---|
| Name | EmaString | | | |
| DictionaryType | Enumeration | • FileDictionary (0)<br>• ChannelDictionary (1) | ChannelDictionary | |
| RdmFieldDictionaryFileName | EmaString | | | Sets the location of the **RdmFieldDictionary** |
| EnumTypeDefFileName | EmaString | | | Sets the location of the **EnumTypeDef** file |

**Table 8: 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"
- **XmlTraceToFile** value="0"

Note that unlike EMA's default behavior of choosing the first **Consumer** component in the **ConsumerList**, EMA applications will not choose the first **Logger**, **Channel**, or **Dictionary** in their respective lists. Instead, if an application wants to use a specific channel, logger, 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**, **Logger**, **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 Scema

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 **Name Consumer_1** which logs to a file named **emaLogfile**.

```
<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>
```

Now assume that the following was not included in **EmaConfig.xml**:

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

In this case, the EMA application relies on its hard-coded behavior and uses the filename **emaLog_*pid*.log**.

However, if the snippet were configured in either of the following configurations, the EMA application would revert to its default behaviors because its parameters are not in the correct order (i.e., the **FileName** parameter needs to be contained in a **Logger** component entry):

- Configuration 1:

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

- Configuration 2:

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

### 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 application ignores **DefaultConsumer**, and the first consumer listed becomes the default.
- If the **EmaConfig.xml** has no **Consumer** components, the EMA application uses the default **Consumer Name** (specified by the **DefaultConsumer** element or, if it is not present, the name **EmaConsumer**).

## 4.3 Configuring EMA Using Function Calls

From an application standpoint, instantiating an OmmConsumerConfig object creates the initial configuration from the **DefaultXML.h** and **EmaConfig.xml** files. 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 |
|---|---|
| username( const EMAstring & ) | Sets the **username** variable. If **username** is not set, the application extracts a username from the run-time environment. |
| password( const EMAstring & ) | Sets the **password** variable. **password** has no default value. |
| position( const EMAstring & ) | Sets the **position** variable. **position** has no default value. |
| applicationId( const EMAstring & ) | Sets the **applicationId** variable. **applicationId** has no default value. |
| host( const EMAstring & ) | Sets the host and port parameters. For details, refer to Section 4.3.2. |
| operationModel( OperationModel ) | Sets the operation model to either `OmmConsumerConfig::ApiDispatchEnum` (which is the default) or `OmmConsumerConfig::UserDispatchEnum`. |
| consumerName ( EmaString & ) | Sets the consumer name. If a consumer does not exist with that name, the application throws an exception. |
| addAdminMsg | Populates part of or all of the login request message, directory request message, or dictionary request message according to the specification discussed in the *RFA C++ Edition Reuters Domain Models (RDM) Usage Guide*. |

**Table 9:**

### 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( EmaString & )` 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_CONN_TYPE_SOCKET**, regardless of any previous setting for that configuration element.

---

**Host** and **Port** values observe the following rules when updating due to the `host( EmaString & )` 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***.

## 4.4 Programmatic Configuration

In addition to changing EMA's configuration via **EmaConfig.xml** or function calls, you can do so programmatically. To do so:

- Create a map with the configuration you want
- Call the `config` method on an OmmConsumerConfig object, and pass the Map as a parameter to the `config` method.

You can pass in multiple maps, each programmatic configuration being applied to create the application's active configuration during instantiation of the OmmConsumer.

# 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_ProcessId.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>
```

THOMSON REUTERS

Document ID: EMA300CG.150
Date of issue: 30 June 2015