

webMethods EntireX

EntireX RPC Server for XML/SOAP

Version 10.8

October 2022

This document applies to webMethods EntireX Version 10.8 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1997-2022 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-XMLRPC-108-20220601

Table of Contents

1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Introduction to the RPC Server for XML/SOAP	5
Administration using Command Central	6
Worker Models	7
3 Administering the RPC Server for XML/SOAP using the Command Central GUI	9
Logging in to Command Central	10
Creating an RPC Server Instance	11
Configuring an RPC Server Instance	16
Viewing the Runtime Status	23
Starting an RPC Server Instance	24
Stopping an RPC Server Instance	26
Inspecting the Log Files	28
Changing the Trace Level Temporarily	29
Deleting an RPC Server Instance	30
4 Administering the RPC Server for XML/SOAP using the Command Central Command Line	31
Creating an RPC Server Instance	32
Configuring an RPC Server Instance	34
Displaying the EntireX Inventory	56
Viewing the Runtime Status	57
Starting an RPC Server Instance	58
Stopping an RPC Server Instance	58
Inspecting the Log Files	59
Changing the Trace Level Temporarily	61
Deleting an RPC Server Instance	62
5 Administering the RPC Server for XML/SOAP with Local Scripts	65
Customizing the RPC Server	66
Configuring the RPC Server	68
Using SOAP 1.2 with the RPC Server	72
Locating and Calling the Target Server	72
Using SSL/TLS with the RPC Server	78
Starting the RPC Server	81
Stopping the RPC Server	81
Pinging the RPC Server	82
Running an EntireX RPC Server as a Windows Service	82
6 Java API	85
Properties File	86
Configuration File	86
Implementation of the Java API for the RPC Server for XML/SOAP	88
Start Script	89

- 7 Running the RPC Server for XML/SOAP in the Software AG Runtime 91
 - Introduction 92
 - Configuration 93
 - Deactivating an RPC Server for XML/SOAP Permanently 93
 - Starting and Stopping the RPC Server for XML/SOAP using JMX (Java Management Extensions) 93
 - Starting and Stopping the Software AG Runtime under UNIX 94
 - Starting and Stopping the Software AG Runtime under Windows 94
- 8 Building an EntireX RPC Server for XML/SOAP Docker Image 97
 - Prerequisites 98
 - Building and Running the RPC Server for XML/SOAP Image 98
 - Verifying the Build 102
 - Healthcheck for RPC Server for XML/SOAP 103

1 About this Documentation

▪ Document Conventions	2
▪ Online Information and Support	2
▪ Data Protection	3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at <https://documentation.softwareag.com>.

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://tech-community.softwareag.com>. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/publishers/softwareag> and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at <https://empower.softwareag.com>. Many services on this portal require that you have an account. If you do not yet have one, you can request it at <https://empower.softwareag.com/register>. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to the RPC Server for XML/SOAP

- Administration using Command Central 6
- Worker Models 7

The EntireX RPC Server for XML/SOAP allows RPC clients to communicate with target servers via HTTP(S). The RPC Server for XML/SOAP transforms RPC client calls into XML/SOAP calls. It works together with the XML/SOAP Wrapper.

Administration using Command Central

Software AG Command Central is a tool that enables you to manage your Software AG products remotely from one location. Command Central offers a browser-based user interface, but you can also automate tasks by using commands to remotely execute actions from a terminal or custom script (for example CI servers such as Jenkins, or generic configuration management tools such as Puppet or Chef).

The screenshot shows the 'Instances' page in the Software AG Command Central interface. The page has a blue header with navigation tabs: 'Installations', 'Stacks', 'Licensing', 'Repositories', 'Jobs', and 'Administrator'. Below the header, there's a breadcrumb trail 'Home > Instances > ALL' and a search bar for environments. The main content area is titled 'Instances' and contains a table of installed components. A message at the top explains how to create and delete instances. The table has columns for Name, Component, Status, Alerts, Installation, and Host. The data rows are:

Name [Count]	Component	Status	Alerts	Installation	Host
EntireX Broker ETB001	EntireX Broker ETB001	Up		Local	localhost
CCE [1 Components]	CCE	Up		Local	localhost
IS_default [3 Components]	IS_default	Up		Local	localhost
SPM [2 Components]	SPM	Up		Local	localhost

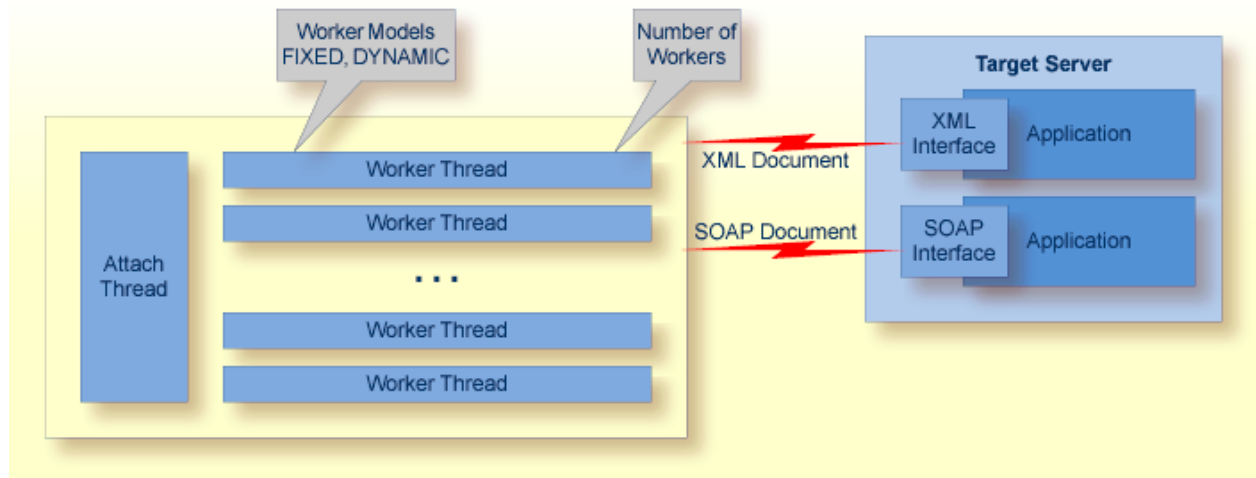
Command Central can assist with the following configuration, management, and monitoring tasks:

- Infrastructure engineers can see at a glance which products and fixes are installed, where they are installed, and compare installations to find discrepancies.
- System administrators can configure environments by using a single web user interface or command-line tool. Maintenance involves minimum effort and risk.
- Release managers can prepare and deploy changes to multiple servers using command-line scripting for simpler, safer lifecycle management.
- Operators can monitor server status and health, as well as start and stop servers from a single location. They can also configure alerts to be sent to them in case of unplanned outages.

The Command Central graphical user interface is described under [Administering the RPC Server for XML/SOAP using the Command Central GUI](#). For the command-line interface, see [Administering the RPC Server for XML/SOAP using the Command Central Command Line](#).

The core Command Central documentation is provided separately and is also available under **Guides for Tools Shared by Software AG Products** on the Software AG documentation website.

Worker Models



RPC requests are worked off inside the RPC server in worker threads. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The RPC Server for XML/SOAP can adjust the number of worker threads to the number of parallel requests. The RPC server provides two worker models:

- **FIXED**

The *fixed* model creates a fixed number of worker threads. The number of worker threads does not increase or decrease during the lifetime of an RPC server instance.

- **DYNAMIC**

The *dynamic* model creates worker threads depending on the incoming load of RPC requests.

For configuration with the Command Central GUI, see [Worker Scalability](#) under *Configuration > Server*.

For technical details, see property `entirex.server.fixedservers` under *Administering the RPC Server for XML/SOAP with Local Scripts*.

3 Administering the RPC Server for XML/SOAP using the Command Central GUI

- Logging in to Command Central 10
- Creating an RPC Server Instance 11
- Configuring an RPC Server Instance 16
- Viewing the Runtime Status 23
- Starting an RPC Server Instance 24
- Stopping an RPC Server Instance 26
- Inspecting the Log Files 28
- Changing the Trace Level Temporarily 29
- Deleting an RPC Server Instance 30

This chapter describes how to administer the EntireX RPC Server for XML/SOAP, using the Command Central graphical user interface.

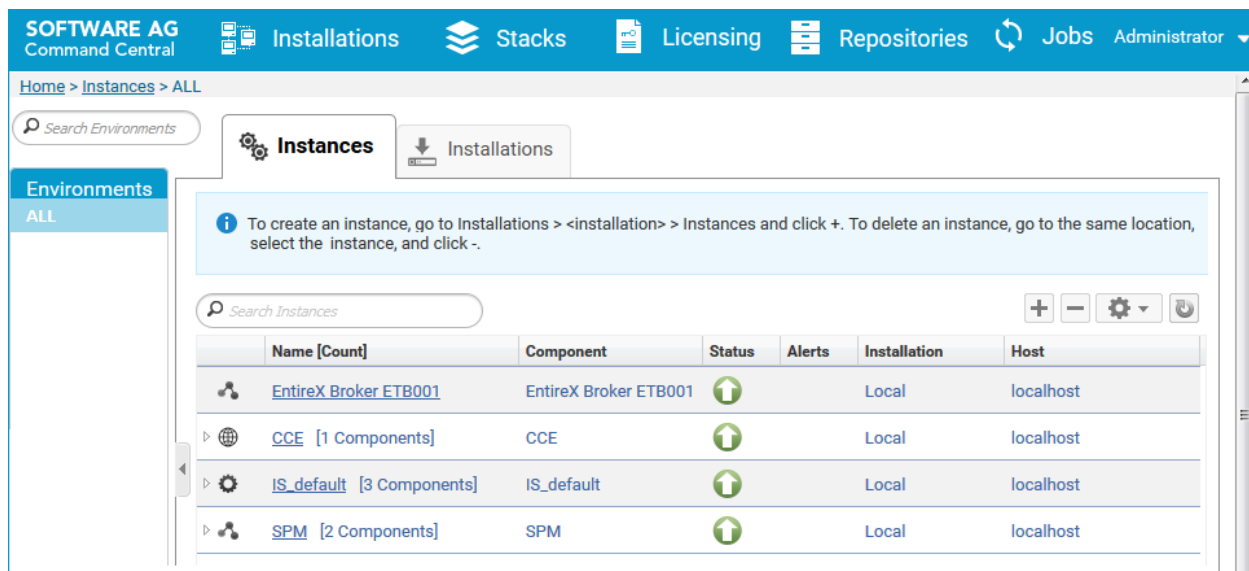
See also [Administering the RPC Server for XML/SOAP using the Command Central Command Line](#). The core Command Central documentation is provided separately and is also available under **Guides for Tools Shared by Software AG Products** on the Software AG documentation website.

Logging in to Command Central

Open an Internet browser and specify the URL of the Command Central Server as follows: `http://<Command_Central_host>:<Command_Central_port>`. This takes you to the Command Central **Login** page.

On Windows you can also get to the **Login** page from the Command Central Start Menu entry.

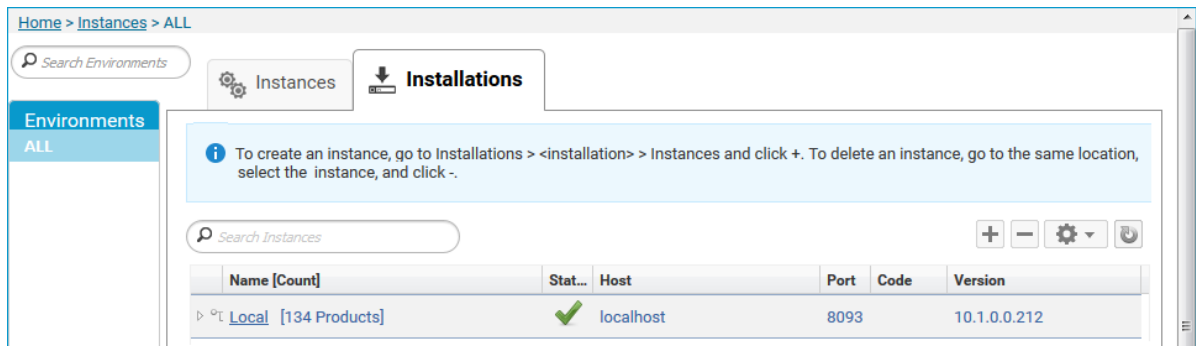
Provide your user credentials in the **Login** page and click **Log In**. This takes you to the page **Home > Instances:**



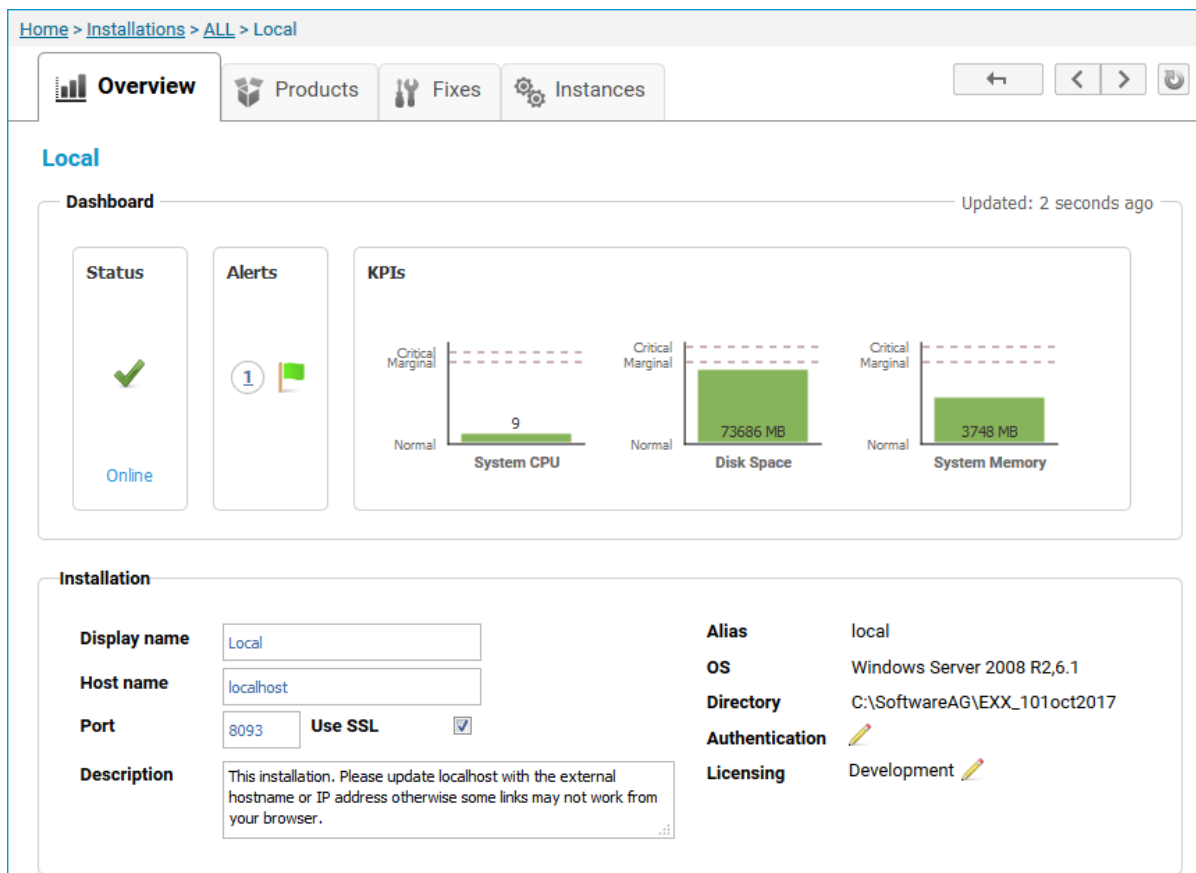
Creating an RPC Server Instance

➤ To create an RPC Server for XML/SOAP instance

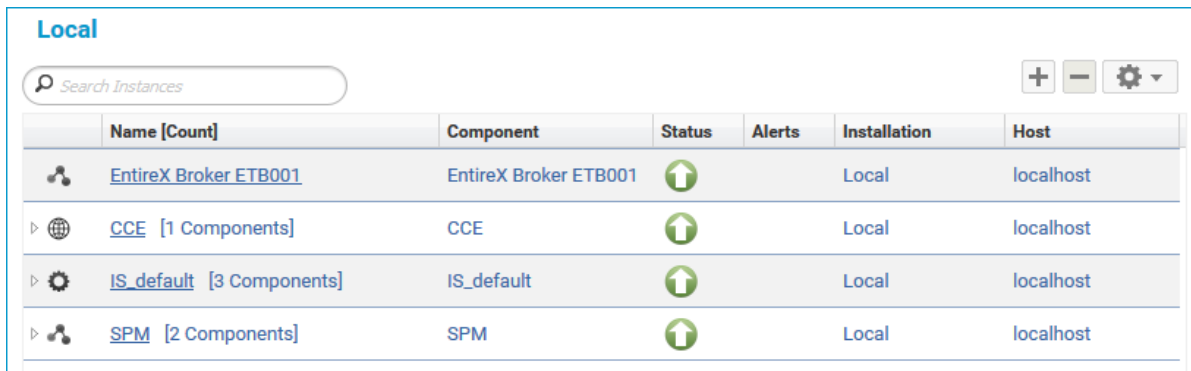
- 1 In the Command Central home page, click the **Installations** tab.



- 2 Click on the desired installation, for example **Local**, where you want to add an RPC Server for XML/SOAP instance.

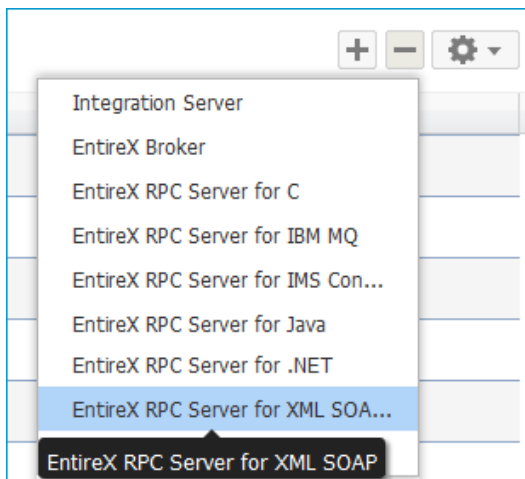


3 Click the **Instances** tab.



	Name [Count]	Component	Status	Alerts	Installation	Host
	EntireX Broker ETB001	EntireX Broker ETB001	↑		Local	localhost
▶	CCE [1 Components]	CCE	↑		Local	localhost
▶	IS_default [3 Components]	IS_default	↑		Local	localhost
▶	SPM [2 Components]	SPM	↑		Local	localhost

4 Click the  button in the upper right corner above the list and choose **EntireX RPC Server for XML/SOAP**.



5 In the **Create Instance** wizard, fill in the fields in the main screen and in the **Server**, **Broker** and **Deployment** tabs.

×
Create Instance - EntireX RPC Server for XML SOAP

1
2

Specify Properties
Summary

i Please specify input parameters for all property tabs (Server, Broker, Deployment)

Instance name *

Register Windows service for automatic startup

Server

Broker

Deployment

Server

RPC Server address * ?

Administration port * ?

Next

Cancel

Main Screen

Parameter	Description
Instance name	Required. Name of the runtime component, for example "MyRpcServer".
Register Windows Service for automatic startup	Optional. Register Windows Service for automatic startup. Default is not checked. If this parameter is checked, the RPC server can be controlled by the Windows Service Control Manager.

Server Tab

Parameter	Description
RPC Server address	Required. The case-sensitive RPC server address has the format: CLASS/SERVER/SERVICE.
Administration port	Required. The administration port in range from 1025 to 65535.

Broker Tab

Parameter	Description
Connection	
Transport	Transport over TCP or SSL. Default is TCP.
Broker host	Required. EntireX Broker host name or IP address. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.
Broker port	Required. Port number in range from 1025 to 65535.
SSL trust store	Optional. Specifies the location of SSL trust store.
Credentials	
User	Optional. The user ID for secured access to the broker.
Password	Optional. The password for secured access to the broker.

Deployment Tab

Here you can enable the RPC Server for XML/SOAP to accept dynamic configuration and deployment of XML mapping files with the Designer.

Parameter	Description
Allow dynamic configuration and deployment of XML Mapping Files (*.xmm).	Default is false.

- 6 Press **Next** to get to the **Summary** page to verify your input.
- 7 Press **Finish**.

The screenshot shows the 'Local' section of the Command Central GUI. At the top, there is a search bar labeled 'Search Instances'. Below it is a table with the following columns: Name [Count], Component, Status, Alerts, Installation, and Host. The table lists several instances, all with a status of 'Up' (indicated by a green arrow icon) and installed on 'localhost'. A modal dialog box titled 'Operation triggered' is overlaid on the table, displaying the message 'Job operation is started successfully.' and two buttons: 'View Job' and 'Finish'.

Name [Count]	Component	Status	Alerts	Installation	Host
EntireX Broker ETB001	EntireX Broker ETB001	Up		Local	localhost
CCE [1 Components]	CCE	Up		Local	localhost
IS_default [3 Components]	IS_default	Up		Local	localhost
SPM [2 Components]	SPM	Up		Local	localhost

The new instance *myRpcServer* appears in the list.

Migrating a Configuration File

After you have created an RPC Server for XML/SOAP using Command Central, you can copy an existing configuration file *entirex.xmlrpcserver.configuration.xml* to your newly created server.

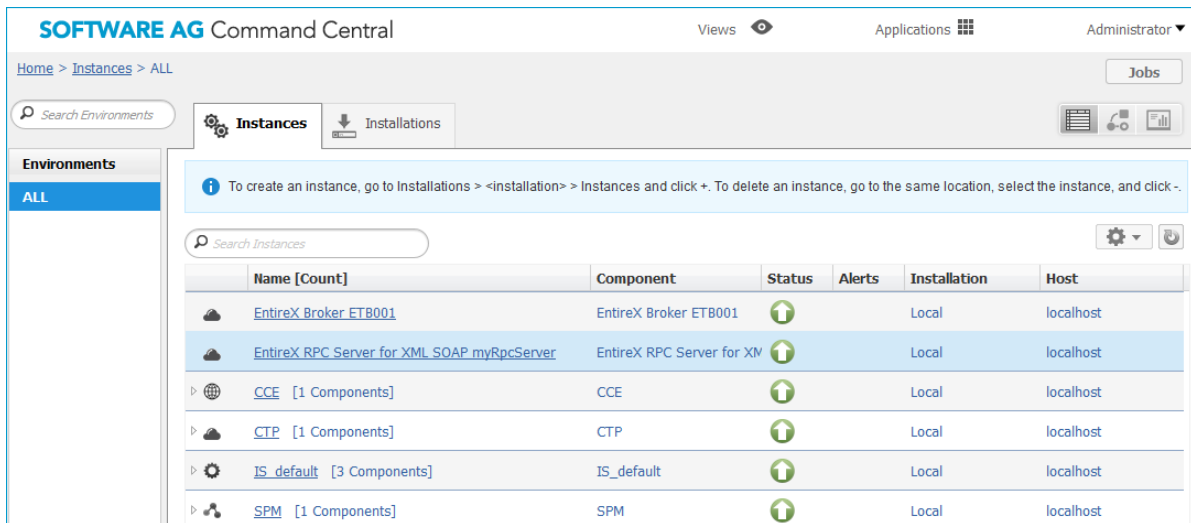
➤ **To migrate an existing configuration file to an RPC Server for XML/SOAP that was created using Command Central**

- 1 Copy file *entirex.xmlrpcserver.configuration.xml* to `<EntireX home>/config/rpc/EntireXCore-RpcServerXml-<instance name>`.
- 2 Remove file *connections.cfg*.
- 3 Rename file *entirex.xmlrpcserver.configuration.xml* to *connections.cfg*.

Configuring an RPC Server Instance

➤ To configure an RPC Server for XML/SOAP instance

- 1 In the Command Central home page, click the **Instances** tab.



- 2 Click on the link associated with this instance to select the RPC server instance you want to configure.

Instance: EntireX RPC Server for XML SOAP myRpcServer Updated: 2 seconds ago

Dashboard

Status

↑

Online

Alerts

0

KPIs

Critical
Marginal

Normal 1

Active Workers

Critical
Marginal

Normal 0

Busy Workers

Details

Display name ⌵

Component EntireX RPC Server for XML SOA...

Host name localhost

Authentication ✎

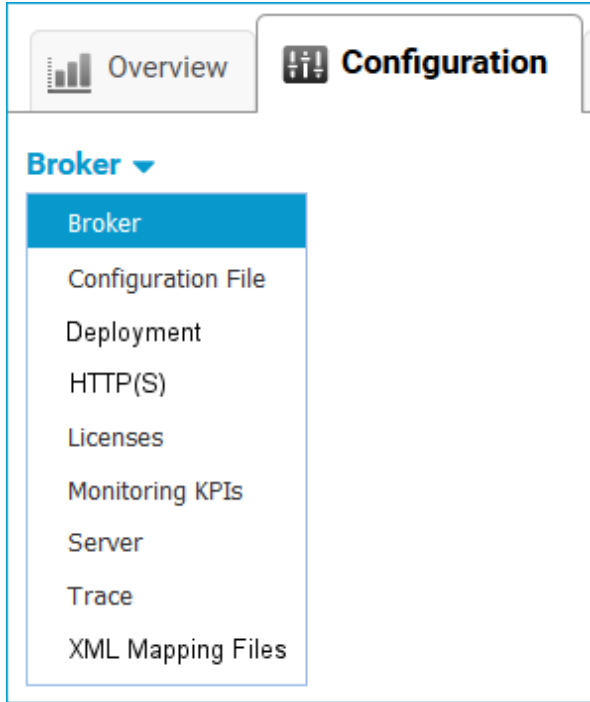
Installation name Local


Installation alias local

Attributes + -

Name	Value

- 3 Click the **Configuration** tab. EntireX supports the following configuration types, which are presented in a drop-down box when you click the down arrow below the **Configuration** tab label:



 **Note:** All configuration changes require a restart of the instance to take effect.

- **Broker**
- **Configuration File**
- **Deployment**
- **HTTP(S)**
- **Licenses**
- **Monitoring KPIs**
- **Server**
- **Trace Level**
- **XML Mapping Files**

Broker

Parameter	Description
Connection	
Transport	Transport over TCP or SSL. Default is TCP.
Broker host	Required. EntireX Broker host name or IP address. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.
Broker port	Required. Port number in range from 1025 to 65535.

Parameter	Description
Encoding	Required. Encoding used for the communication between the RPC server and EntireX Broker.
SSL trust store	Optional. Specifies the location of SSL trust store.
SSL verify server	Optional. The RPC server as SSL client checks the identity of the broker as SSL server.
FIPS-140 mode	Optional. Enable FIPS-140 compliant SSL communication. Default is <code>no</code> .
Credentials	
User	Optional. The user ID for secured access to the broker.
Password	Optional. The password for secured access to the broker.

Configuration File

Here you can view/edit the configuration file of the RPC Server for XML/SOAP.

Deployment

Here you can enable the RPC Server for XML/SOAP to accept dynamic configuration and deployment of XML mapping files with Software AG designer.

Parameter	Description
Allow dynamic configuration and deployment of XML Mapping Files (*.xmm).	Default is <code>false</code> .

HTTP(S)

Here you can modify the HTTP(S) parameters of the RPC Server for XML/SOAP.

Parameter	Description
Proxy Setting for HTTP	
Host	Optional. The host name or IP address of the proxy server for HTTP connection.
Port	Optional. The port number of the proxy server for HTTP connection.
User	Optional. The user name of the proxy server for HTTP connection.
Password	Optional. The password of the proxy server for HTTP connection.
Proxy Setting for HTTPS	
Host	Optional. The host name or IP address of the proxy server for HTTPS connection.
Port	Optional. The port number of the proxy server for HTTPS connection.
User	Optional. The user name of the proxy server for HTTPS connection.
Password	Optional. The password of the proxy server for HTTPS connection.
SSL Trust Store	
Location	Optional. Specifies the location of the SSL trust store used for HTTPS connection.
Non Proxy Hosts	

Parameter	Description
Host name	Optional. List of the hosts that should be accessed without going through the proxy. Asterisk notation is allowed.

Licenses

Here you can view/set the license file in the EntireX installation. For details see *Point to the License Key for an Instance or Component* under *Working with Standalone Product Installation* in the Command Central documentation.



Note: The license file is used for all EntireX instances in this installation.

Monitoring KPIs

Here you can modify margins of monitored key performance indicators (KPIs) available for the RPC Server for XML/SOAP: Active Workers and Busy Workers.

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for XML/SOAP. The following KPIs help you administer, troubleshoot, and resolve performance issues:

KPI	Setting
Absolute number of Active Workers	entirex.generic.kpi.1.max=20
Critical alert relative to maximum	entirex.generic.kpi.1.critical=0.95
Marginal alert relative to maximum	entirex.generic.kpi.1.marginal=0.80
Absolute number of Busy Workers	entirex.generic.kpi.2.max=20
Critical alert relative to maximum	entirex.generic.kpi.2.critical=0.95
Marginal alert relative to maximum	entirex.generic.kpi.2.marginal=0.80

Do not change the other properties!

Server

Here you can specify the RPC Server settings.

Parameter	Description
RPC Server	
RPC Server address	Required. The case-sensitive RPC server address has the format: CLASS/SERVER/SERVICE.
Administration port	Required. The administration port in range from 1025 to 65535.
Reconnection attempts	Required. Number of reconnection attempts to the broker. When the number of attempts is reached and a connection to the broker is not possible, the RPC Server for XML/SOAP stops.

Parameter	Description
Worker Scalability	
Worker model	You can either have a fixed or dynamic number of workers. Default is dynamic (true). For more information see Worker Models .
Fixed number	Required. Fixed number of workers. Must be a number in range from 1 to 255.
Minimum number	Required. Minimum number of workers. Must be a number in range from 1 to 255.
Maximum number	Required. Maximum number of workers. Must be a number in range from 1 to 255.

Trace Level

Here you can set the trace level of the RPC Server for XML/SOAP.

Parameter	Value	Description
Trace level	0 - 3	One of the following levels: 0 - None - No trace output (default). 1 - Standard - Minimal trace output. 2 - Advanced - Detailed trace output. 3 - Support - Support diagnostic. Use only when requested by Software AG Support.

XML Mapping Files

Here you can show the list of XML mapping files configured for this RPC server. You can add/delete XML mapping files or modify the settings for a listed XML mapping file.

Parameter	Description
Mapping	
XML Mapping file	Required. Absolute location including name of XML mapping file.
Web service connection URL	Required. URL of the web service to call.
Time Out (sec.)	HTTP(S) connection timeout (seconds).
Soap version	
SOAP version	RPC Server for XML/SOAP communicates with SOAP version 1.1 or 1.2. Note: Mapping using pure XML is not affected by this setting.
Web Service Transport Security	
Basic authentication	Disable/enable the basic authentication for web service connection. No authentication Disable basic authentication. For defined user/password The basic authentication uses the credentials defined for user and password

Parameter	Description
	<p>Using RPC client credentials The basic authentication uses the credentials set by RPC client application.</p> <ul style="list-style-type: none"> ■ For how to send the RPC user ID/password pair from an RPC client, see <i>Using the Broker and RPC User ID/Password</i> (.NET Wrapper Java Wrapper C Wrapper PL/I Wrapper DCOM Wrapper Web Services Wrapper IDL Tester Listener for XML/SOAP Listener for IBM MQ). ■ For the COBOL Wrapper, refer to <i>Using Broker Logon and Logoff</i> and <i>Using RPC Authentication (Natural Security, Impersonation, Integration Server)</i>. ■ For non-RPC clients, see <i>Using the Broker and RPC User ID/Password</i> under <i>EntireX XML Tester</i> in the XML/SOAP Wrapper documentation.
User	User for basic authentication.
Password	Password for basic authentication.
Web Service Message Security	
UsernameToken type	<p>Authenticate user with UsernameToken security.</p> <p>No UsernameToken security Payload does not contains a UsernameToken.</p> <p>The password in plain text The UsernameToken in payload contains the plain text password. We strongly recommend secure transport (HTTPS) when sending the password in plain text.</p> <p>The digest of the password The UsernameToken in payload contains digest of the password.</p>
WSDL for Advanced Configuration of Web Service e.g. Schema Validation or WS Policy	
WsdL file	Optional. Absolute location including name of WSDL file. If you are using a WSDL file, the address of the web service is retrieved from this.
WsdL service	Optional. The service name to use must be defined in WSDL file and match the selected SOAP version. If the service name is unique for the SOAP version, it can be omitted.
WsdL port	Optional. The port name to use must be defined in WSDL file and match the selected SOAP version. If the port name is unique for the SOAP version, it can be omitted.

- 4 Click **Edit** to modify the parameters on your selected configuration type.
- 5 Click **Test** to check the correctness of your input or **Apply** to save your changes.

Viewing the Runtime Status

➤ To view the runtime status of the RPC server instance

- In the Command Central **Home** page, click the **Instances** tab and select the RPC Server for XML/SOAP instance for which you want to see the runtime status (same as Step 1 under *Configuring a Broker Instance*).



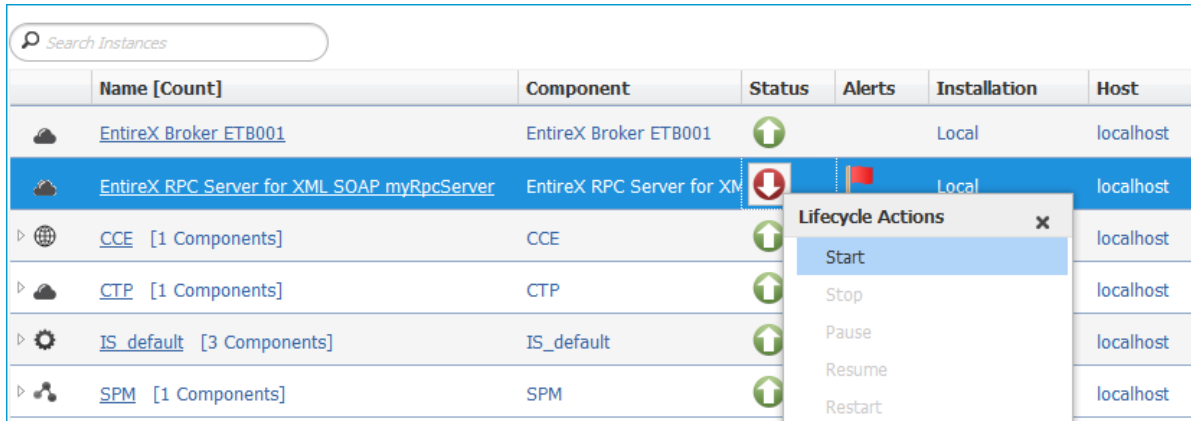
The visual key performance indicators (KPIs) and alerts enable you to monitor the RPC Server for XML/SOAP's health.

KPI	Description
Active Workers	Number of active workers.
Busy Workers	Number of busy workers.

Starting an RPC Server Instance

➤ **To start an RPC Server for XML/SOAP instance from the Instances tab**

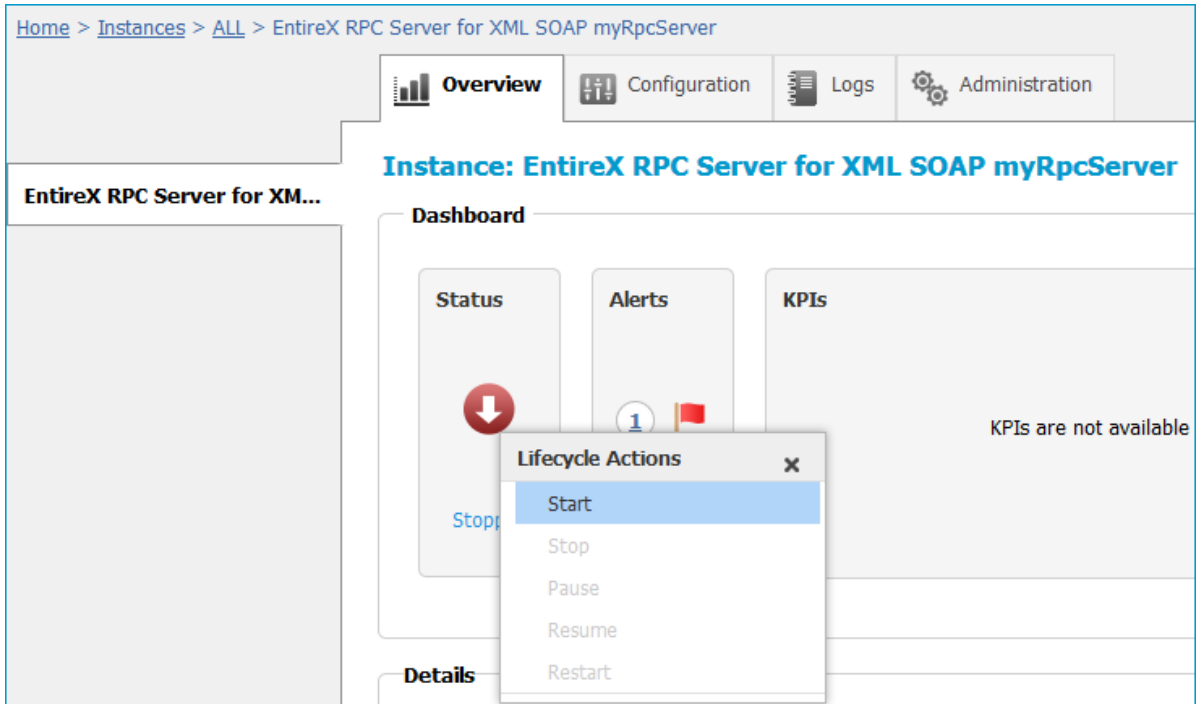
- 1 In the Command Central home page, click the **Instances** tab.



- 2 Select the status, and from the context menu choose **Start**.

➤ **To start an RPC Server for XML/SOAP instance from its Overview tab**

- 1 In the Command Central home page, click the **Instances** tab and select the RPC Server for XML/SOAP instance you want to start (same as Step 1 under *Configuring a Broker Instance*).

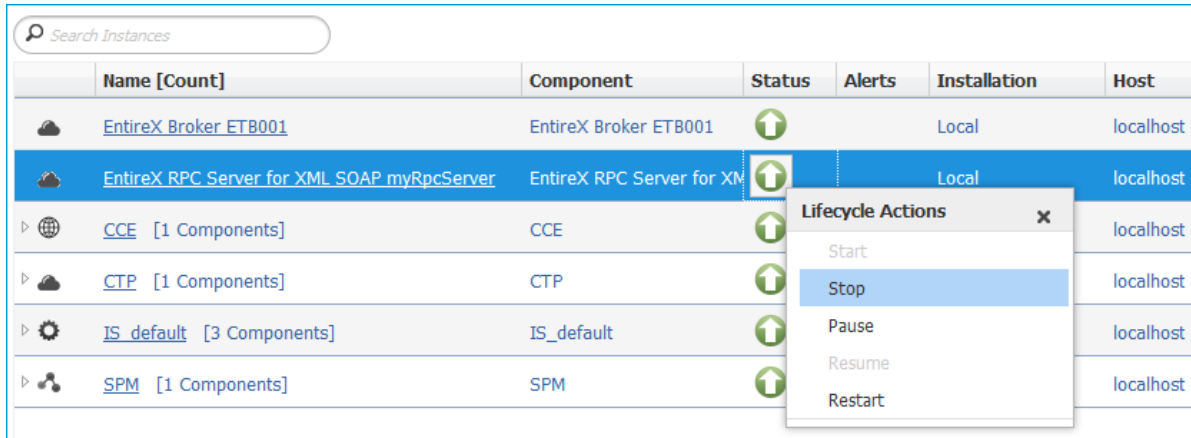


- 2 Select the status, and from the context menu choose **Start**.

Stopping an RPC Server Instance

> **To stop an RPC Server for XML/SOAP instance from the Instances tab**

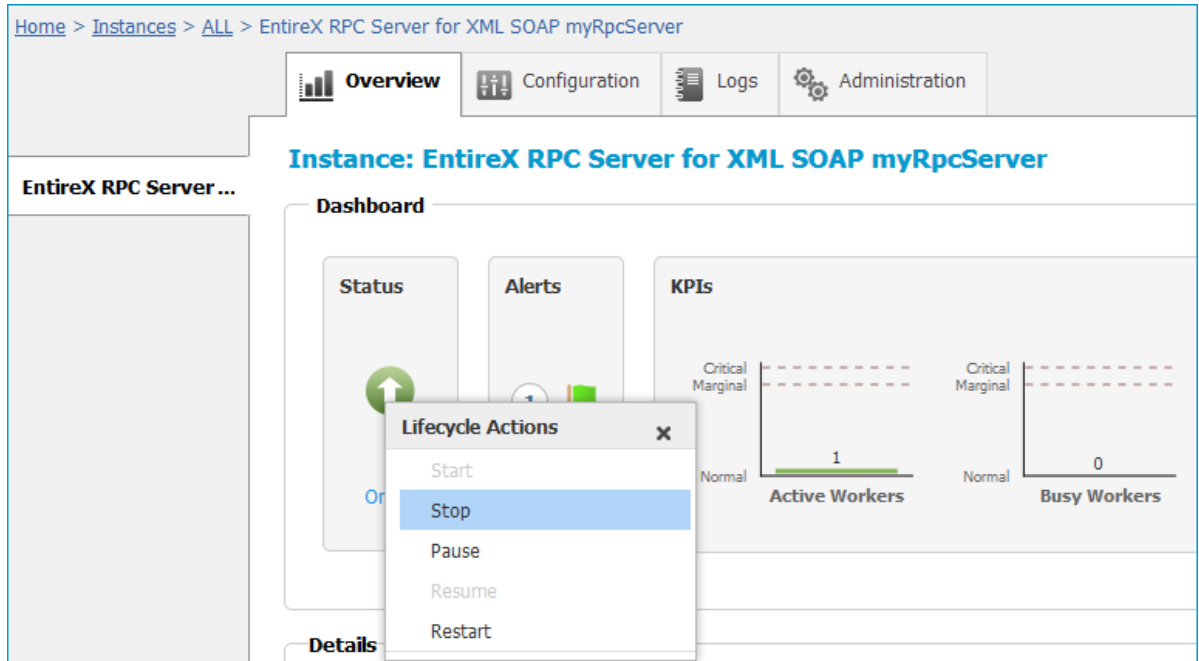
- 1 In the Command Central home page, click the **Instances** tab.



- 2 Select the status, and from the context menu choose **Stop**.

> **To stop an RPC Server for XML/SOAP instance from its Overview tab**

- 1 In the Command Central home page, click the **Instances** tab and select the RPC Server for XML/SOAP instance you want to stop (same as Step 1 under *Configuring a Broker Instance*).

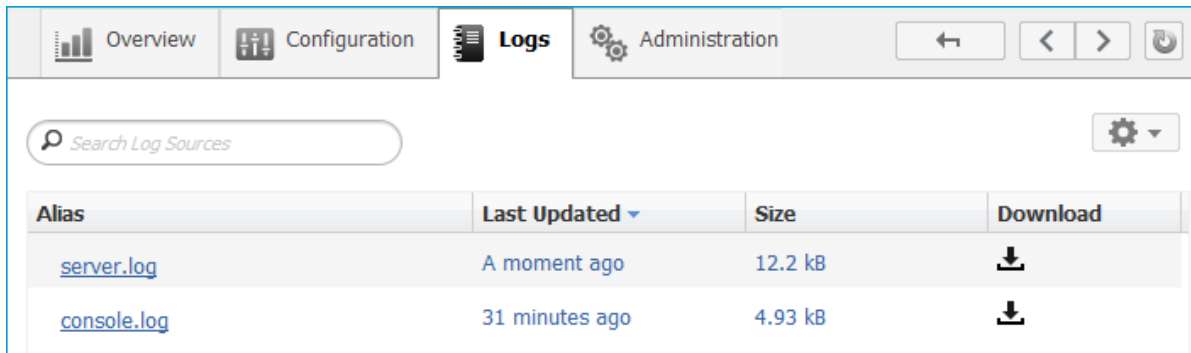


- 2 Select the status, and from the context menu choose **Stop**.

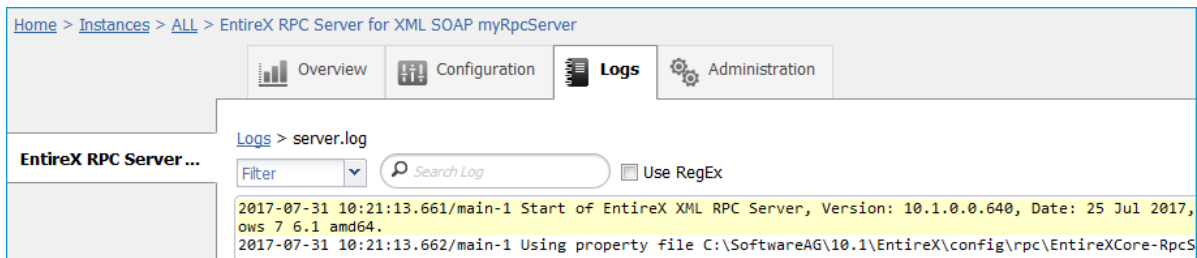
Inspecting the Log Files

➤ To inspect the log files of an RPC Server for XML/SOAP instance

- 1 In the Command Central home page, click the **Instances** tab, then click the link associated with the RPC Server for XML/SOAP instance for which you want to inspect the log files (same as Step 1 under *Configuring a Broker Instance*).
- 2 Click the **Logs** tab:



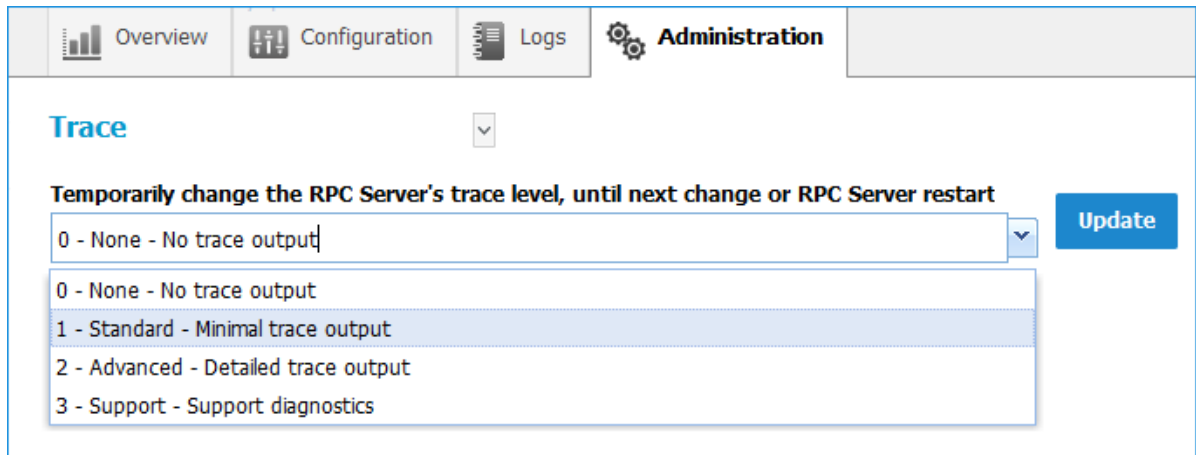
- 3 In the **Alias** column, click the link of the log file you want to inspect, for example *server.log*:



Changing the Trace Level Temporarily

➤ To temporarily change the trace level of an RPC Server for XML/SOAP instance


- 1 In the Command Central home page, click the **Instances** tab then click the link associated with the RPC Server for XML/SOAP instance for which you want change the trace level temporarily (same as Step 1 under *Configuring a Broker Instance*).
- 2 In the **Administration** tab, select the trace level and press **Update**.

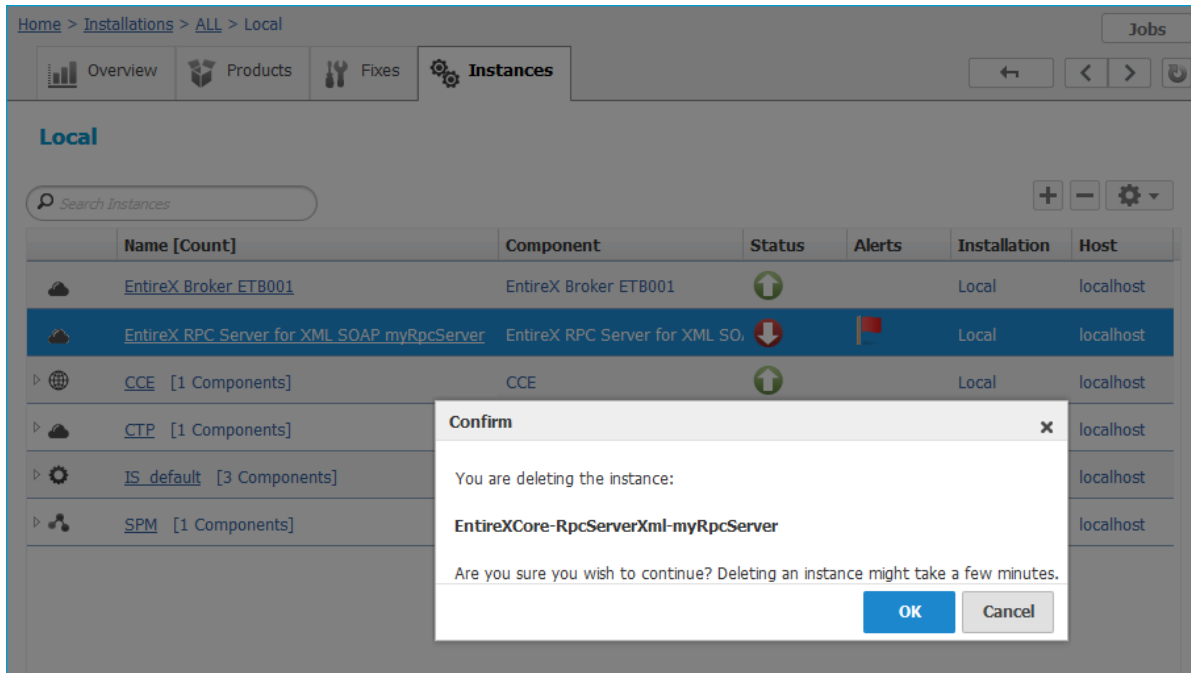


Note: If you want to set the trace level permanently, see [Trace Level](#) under *Configuring an RPC Server Instance*.

Deleting an RPC Server Instance

➤ To delete an RPC Server for XML/SOAP instance

- 1 In the list of EntireX RPC Server for XML/SOAP instances for your selected installation (for example Local), select the instance you want to delete and click the  button in the upper right corner above the list.



- 2 Click **OK** to confirm the uninstall of this RPC Server for XML/SOAP instance.
- 3 In the next window, click **Finish**. The selected instance is removed from the list.

4 Administering the RPC Server for XML/SOAP using the Command Central Command Line

- Creating an RPC Server Instance 32
- Configuring an RPC Server Instance 34
- Displaying the EntireX Inventory 56
- Viewing the Runtime Status 57
- Starting an RPC Server Instance 58
- Stopping an RPC Server Instance 58
- Inspecting the Log Files 59
- Changing the Trace Level Temporarily 61
- Deleting an RPC Server Instance 62

This chapter describes how to administer the EntireX RPC Server for XML/SOAP, using the Command Central command-line interface.

Administering the RPC Server for XML/SOAP using the Command Central GUI is described under [Administering the RPC Server for XML/SOAP using the Command Central GUI](#). The core Command Central documentation is provided separately and is also available under **Guides for Tools Shared by Software AG Products** on the Software AG documentation website.

Creating an RPC Server Instance

The following table lists the parameters to include when creating an EntireX RPC instance, using the Command Central `create instances` commands.

Command	Parameter	Value	Description
sagcc create instances	<i>node_alias</i>	<i>name</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>type</i>	RpcServerXml	Required. EntireXCore instance type of RPC server. Must be "RpcServerXml".
	<i>product</i>	EntireXCore	Required. Must be set to "EntireXCore".
	<i>instance.name</i>	<i>name</i>	Required. Name of the runtime component, for example "MyRpcServer".
	<i>install.service</i>	true <u>false</u>	Optional. Register Windows Service for automatic startup. Default is false. If this parameter is true, the RPC server can be controlled by the Windows Service Control Manager.
	<i>server.address</i>	<i>class/server/service</i>	Required. The case-sensitive RPC server address has the format: CLASS/SERVER/SERVICE.
	<i>server.adminport</i>	1025-65535	Required. The administration port in range from 1025 to 65535.
	<i>broker.transport</i>	ssl <u>tcp</u>	Transport over TCP or SSL. Default is TCP.
	<i>broker.host</i>	<i>name</i>	Required. EntireX Broker host name or IP address. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.
	<i>broker.port</i>	1025-65535	Required. Port number in range from 1025 to 65535.
	<i>broker.user</i>	<i>user</i>	Optional. The user ID for secured access to the broker.
	<i>broker.password</i>	<i>password</i>	Optional. The password for secured access to the broker.

Example

- To create a new instance for an installed EntireX of the type "RpcServerXml", with name "MyRpcServer", with server address "RPC/SRV1/CALLNAT", using administration port 5757, with broker host name "localhost", listening on broker port 1971, in the installation with alias name "local":

```
sagcc create instances local EntireXCore type=RpcServerXml
instance.name=MyRpcServer server.address=RPC/SRV1/CALLNAT server.adminport=5757
broker.host=localhost broker.port=1971
```

Information about the creation job - including the job ID - is displayed.

Migrating a Configuration File

After you have created an RPC Server for XML/SOAP using Command Central, you can copy an existing configuration file *entirex.xmlrpcserver.configuration.xml* to your newly created server.

➤ To migrate an existing configuration file to an RPC Server for XML/SOAP that was created using Command Central

- 1 Copy file *entirex.xmlrpcserver.configuration.xml* to *<EntireX home>/config/rpc/EntireXCore-RpcServerXml-<instance name>*.
- 2 Remove file *connections.cfg*.
- 3 Rename file *entirex.xmlrpcserver.configuration.xml* to *connections.cfg*.

Configuring an RPC Server Instance

Here you can administer the parameters of the RPC Server for XML/SOAP. Any changes to parameters will be used the next time you start the RPC server.

- [Broker](#)
- [Configuration File](#)
- [Deployment](#)
- [HTTP\(S\)](#)
- [Monitoring KPIs](#)
- [Server](#)
- [Trace](#)
- [XML Mapping Files](#)

Broker

Here you can administer the parameters used for communication between the RPC Server for XML/SOAP and EntireX Broker.

- [Parameters](#)
- [Displaying the Broker Settings of the RPC Server](#)
- [Updating the Broker Settings of the RPC Server](#)

Parameters

Parameter	Value	Description
BrokerTransport	TCP SSL	Transport over TCP or SSL. Default is TCP.
BrokerHost	name	Required. EntireX Broker host name or IP address. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.
BrokerPort	1025-65535	Required. Port number in range from 1025 to 65535.
BrokerUser	user	Optional. The user ID for secured access to the broker.
BrokerPassword	password	Optional. The password for secured access to the broker.
BrokerEncoding	codepage	Required. Encoding used for the communication between the RPC server and EntireX Broker.
BrokerSslTrustStore	filename	Optional. Specifies the location of SSL trust store.
BrokerSslVerifyServer	true false	Optional. The RPC server as SSL client checks the identity of the broker as SSL server.
BrokerFipsMode	yes no	Optional. Enable FIPS-140 compliant SSL communication. Default is no.

Displaying the Broker Settings of the RPC Server

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "BROKER".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the Broker parameters of the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer BROKER
```

Example 2

- To store the Broker parameters in the file *broker.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer BROKER -o broker.json
```

Resulting output file in JSON format:

```
{
  "BrokerHost": "localhost",
  "BrokerPort": "1971",
  "BrokerTransport": "TCP",
  "BrokerUser": "testuser",
  "BrokerPassword": "",
  "BrokerEncoding": "Cp1252",
  "BrokerSslTrustStore": "",
  "BrokerSslVerifyServer": "true"
  "BrokerFipsMode": "no"
}
```

Updating the Broker Settings of the RPC Server

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "BROKER".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the Broker parameters of the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" from the file *broker.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer BROKER
-i broker.json
```

See [Example 2](#) above for sample input file.

Configuration File

Here you can administer the configuration file of the RPC Server for XML/SOAP. Any changes will take effect after the next restart.

- [Displaying the Content of the RPC Server Configuration File](#)
- [Updating the Content of the RPC Server Configuration File](#)

Displaying the Content of the RPC Server Configuration File

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "CONFIGURATION".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the configuration file of the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer
CONFIGURATION
```

Example 2

- To store the contents of the configuration file in the text file *configuration.txt* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer
CONFIGURATION -o configuration.txt
```

Updating the Content of the RPC Server Configuration File

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "CONFIGURATION".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the contents of configuration file *configuration.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer  
CONFIGURATION -i configuration.json
```

Deployment

Here you modify the parameter to allow dynamic deployment of XML mapping files.

- [Parameter](#)
- [Displaying the Deployment Parameter](#)
- [Updating the Deployment Parameter](#)

Parameter

Parameter	Value	Description
Enabled	true false	A running RPC Server for XML/SOAP accepts/rejects XML mapping file (XMM) deployment. Default is rejects (false).

Displaying the Deployment Parameter

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "DEPLOYMENT".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the Deployment parameter of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer DEPLOYMENT
```

Example 2

- To store the Deployment parameter in the file *deployment.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer DEPLOYMENT -o deployment.json
```

Resulting output file in JSON format:

```
{
  "Enabled": "true"
}
```

Updating the Deployment Parameter

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "DEPLOYMENT".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the Deployment parameter in the installation with alias name "local" from file *deployment.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer
DEPLOYMENT -i deployment.json
```

See [Example 2](#) above for sample output file.

HTTP(S)

Here you can modify the HTTP(S) parameters of the RPC Server for XML/SOAP.

- [Parameters](#)
- [Displaying the HTTP\(S\) Parameters](#)
- [Updating the HTTP\(S\) Parameters](#)

Parameters

Parameter	Description
HttpProxyHost	The host name or IP address of the proxy server for HTTP connection.
HttpProxyPort	The port number of the proxy server for HTTP connection.
HttpProxyUser	The user name for proxy authentication for HTTP connection if required.
HttpProxyPassword	The password for proxy authentication for HTTP connection if required.
HttpsProxyHost	The host name or IP address of the proxy server for HTTPS connection.
HttpsProxyPort	The port number of the proxy server for HTTPS connection.
HttpsProxyUser	The user name for proxy authentication for HTTPS connection if required.
HttpsProxyPassword	The password for proxy authentication for HTTPS connection if required.
TrustStoreLocation	Specifies the location of SSL trust store used for HTTPS connection.
HttpNonProxyHosts	List of the hosts that should be accessed without going through the proxy. Asterisk notation allowed.

Displaying the HTTP(S) Parameters

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "HTTPS".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the HTTP(S) parameters of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer HTTPS
```

Example 2

- To store the server parameters in the file *http.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer HTTPS -o http.json
```

Resulting output file in JSON format:

```
{
  "HttpProxyHost": "httpHost",
  "HttpProxyPort": "11111",
  "HttpProxyUser": "httpUser",
  "HttpProxyPassword": "httpPassword",
  "HttpsProxyHost": "sslHost",
  "HttpsProxyPort": "22222",
  "HttpsProxyUser": "sslUser",
  "HttpsProxyPassword": "sslPassword",
  "TrustStoreLocation": "c:/Truststore.jks",
  "HttpNonProxyHosts": [
    {
      "HostPattern": "anyhost"
    },
    {
      "HostPattern": "anyhost2"
    }
  ]
}
```

Updating the HTTP(S) Parameters

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "HTTPS".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the HTTP(S) parameters in the installation with alias name "local" from file *http.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer HTTPS  
-i http.json
```

See [Example 2](#) above for sample output file.

Monitoring KPIs

Here you can administer margins of monitored key performance indicators (KPIs) available for the RPC Server for XML/SOAP: Active Workers and Busy Workers.

- [Parameters](#)
- [Displaying the Monitoring KPIs](#)
- [Updating the Monitoring KPIs](#)

Parameters

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for XML/SOAP. The following KPIs help you administer, troubleshoot, and resolve performance issues:

KPI	Setting
Absolute number of Active Workers	entirex.generic.kpi.1.max=20
Critical alert relative to maximum	entirex.generic.kpi.1.critical=0.95
Marginal alert relative to maximum	entirex.generic.kpi.1.marginal=0.80
Absolute number of Busy Workers	entirex.generic.kpi.2.max=20
Critical alert relative to maximum	entirex.generic.kpi.2.critical=0.95
Marginal alert relative to maximum	entirex.generic.kpi.2.marginal=0.80

Do not change the other properties!

Displaying the Monitoring KPIs

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "EXX-MONITORING-KPIS".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the monitoring KPI properties of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:


```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer
MONITORING-KPI
```

Example 2

- To store the monitoring KPI properties in the file *my.properties* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer
MONITORING-KPI -o my.properties
```

Resulting output file in text format:

```
entirex.entirex.spm.version=10.8.0.0.473
entirex.generic.kpi.1.critical=0.95
entirex.generic.kpi.1.id=\#1
entirex.generic.kpi.1.marginal=0.80
entirex.generic.kpi.1.max=20
entirex.generic.kpi.1.name=Active Workers
entirex.generic.kpi.1.unit=
entirex.generic.kpi.1.value=0
entirex.generic.kpi.2.critical=0.95
entirex.generic.kpi.2.id=\#2
entirex.generic.kpi.2.marginal=0.80
entirex.generic.kpi.2.max=20
entirex.generic.kpi.2.name=Busy Workers
entirex.generic.kpi.2.unit=
entirex.generic.kpi.2.value=0
```

Updating the Monitoring KPIs

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "EXX-MONITORING-KPIS".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the contents of file *my.properties* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer  
MONITORING-KPI -i my.properties
```

Server

Here you can administer the parameters defining the registration name, the administration port and the behavior of the RPC Server for XML/SOAP.

- [Parameters](#)
- [Displaying the Server Settings](#)
- [Updating the Server Settings](#)

Parameters

Parameter	Value	Description
ServerAddress	<i>class/server/service</i>	Required. The case-sensitive RPC server address has the format: CLASS/SERVER/SERVICE.
ServerAdminport	1025-65535	Required. The administration port in range from 1025 to 65535.
ReconnectionAttempts	<i>n</i>	Required. Number of reconnection attempts to the broker. When the number of attempts is reached and a connection to the broker is not possible, the RPC Server for XML/SOAP stops.
WorkerScalability	<i>true</i> <i>false</i>	You can either have a fixed or dynamic number of workers. Default is dynamic (<i>true</i>). For more information see Worker Models .
FixNumber	1-255	Required. Fixed number of workers. Must be a number in range from 1 to 255.
MinWorkers	1-255	Required. Minimum number of workers. Must be a number in range from 1 to 255.
MaxWorkers	1-255	Required. Maximum number of workers. Must be a number in range from 1 to 255.

Displaying the Server Settings

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "SERVER".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the server parameters of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer SERVER
```

Example 2

- To store the server parameters in the file *server.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer SERVER
-o server.json
```

Resulting output file in JSON format:

```
{
  "ServerAddress": "RPC/SRV1/CALLNAT",
  "ServerAdminport": "4711",
  "ReconnectionAttempts": "15",
  "WorkerScalability": "true",
  "FixNumber": "5",
  "MinWorkers": "1",
  "MaxWorkers": "10"
}
```

Updating the Server Settings

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "SERVER".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the server parameters from the file *server.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer SERVER  
-i server.json
```

See [Example 2](#) above for sample input file.

Trace

Here you can set the trace level of the RPC Server for XML/SOAP.

- [Parameters](#)
- [Displaying the Trace Level](#)
- [Updating the Trace Level](#)

Parameters

Parameter	Value	Description
TraceLevel	0 1 2 3	One of the following levels: 0 - None - No trace output (default). 1 - Standard - Minimal trace output. 2 - Advanced - Detailed trace output. 3 - Support - Support diagnostic. Use only when requested by Software AG Support.

Displaying the Trace Level

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "TRACE".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the trace level of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer TRACE
```

Example 2

- To store the trace level in the file *trace.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer TRACE -o
trace.json
```

Resulting output file in JSON format:

```
{
  "TraceLevel": "0"
}
```

Updating the Trace Level

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "TRACE".
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the trace level parameters from the file *trace.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer TRACE
-i trace.json
```

See [Example 2](#) above for sample input file.

XML Mapping Files

Here you can add, modify or remove the XML mapping files and the settings that the RPC Server for XML/SOAP uses to connect to a web service.

- [Parameters](#)
- [Listing the XML/SOAP Mapping Files](#)
- [Creating an XML/SOAP Mapping File](#)
- [Displaying an XML/SOAP Mapping File](#)

■ Updating an XML/SOAP Mapping File

Parameters

Parameter	Value	Description
Id	<i>instanceid</i>	The instance ID of the XML mapping file. Required when updating an XML/SOAP mapping file; must be omitted for creating a new XML/SOAP mapping file.
ConnectionUrl	<i>url</i>	Required. URL of the web service to call.
TimeOut	<i>n</i>	HTTP(S) connection timeout (seconds).
XmmFile	<i>filename</i>	Required. Absolute location including name of XML mapping file.
SoapVersion	<u>soap1</u> soap2	RPC Server for XML/SOAP communicates with SOAP version 1.1 or 1.2. Note: Mapping using pure XML is not affected by this setting.
WsdFile	<i>filename</i>	Optional. Absolute location including name of WSDL file. If you are using a WSDL file, the address of the web service is retrieved from this.
WsdService	<i>servicename</i>	Optional. The service name to use must be defined in WSDL file and match the selected SOAP version. If the service name is unique for the SOAP version, it can be omitted.
WsdPort	<i>portname</i>	Optional. The port name to use must be defined in WSDL file and match the selected SOAP version. If the port name is unique for the SOAP version, it can be omitted.
BasicAuthentication	disabled baFixed baRuntime	Disable/enable the basic authentication for web service connection. disabled Disable basic authentication. baFixed The basic authentication uses the credentials defined for user and password baRuntime The basic authentication uses the credentials set by RPC client application. ■ For how to send the RPC user ID/password pair from an RPC client, see <i>Using the Broker and RPC User ID/Password</i> (.NET Wrapper Java Wrapper C Wrapper PL/I Wrapper DCOM Wrapper Web Services Wrapper IDL Tester Listener for XML/SOAP Listener for IBM MQ). ■ For the COBOL Wrapper, refer to <i>Using Broker Logon and Logoff</i> and <i>Using RPC Authentication</i>

Parameter	Value	Description
		<p>(Natural Security, Impersonation, Integration Server).</p> <ul style="list-style-type: none"> For non-RPC clients, see <i>Using the Broker and RPC User ID/Password</i> under <i>EntireX XML Tester</i> in the XML/SOAP Wrapper documentation.
AuthUser	<i>user</i>	User for basic authentication.
AuthPassword	<i>password</i>	Password for basic authentication.
UsernameToken	disabled passwordText passwordDigest	<p>Authenticate user with UsernameToken security.</p> <p>disabled Payload does not contains a UsernameToken.</p> <p>passwordText The UsernameToken in payload contains the plain text password. We strongly recommend secure transport (HTTPS) when sending the password in plain text.</p> <p>passwordDigest The UsernameToken in payload contains digest of the password.</p>

Listing the XML/SOAP Mapping Files

Command	Parameter	Description
sagcc list configuration instances local	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example

- To display the XML/SOAP mappings of instances list:

```
sagcc list configuration instances local EntireXCore-RpcServerXml-MyRpcServer
```

The mapping files have the prefix "XMM-".

Creating an XML/SOAP Mapping File

Command	Parameter	Description
sagcc create configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. Must be "MAPPING"
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To create an XML/SOAP mapping entry for RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" from file *newMapping.json* in the current working directory, the *newMapping.json* file must *not* contain an ID:

```
sagcc create configuration data local EntireXCore-RpcServerXml-MyRpcServer MAPPING
-i newMapping.json
```

Displaying an XML/SOAP Mapping File

Command	Parameter	Description
sagcc get configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. The instanceid of an XML/SOAP mapping is prefixed with "XMM-" followed by a number.
	<i>-o file</i>	Optional. Specifies the file where you want the output written.

Example 1

- To display the parameters of XML/SOAP mapping with instance ID "XMM-1" of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer XMM-1
```

Example 2

- To store the parameters of XML/SOAP mapping with instance ID "XMM-1" in the file *mapping1.json* in the current working directory:

```
sagcc get configuration data local EntireXCore-RpcServerXml-MyRpcServer XMM-1 -o mapping1.json
```

Resulting output file in JSON format:

```
{
  "Id": "XMM-1",
  "ConnectionUrl": "http://www.sample.sample/greeting",
  "Timeout": "33",
  "XmmFile": "c:/xmlserver/sample.xmm",
  "SoapVersion": "soap11",
  "Wsd1File": "c:/xmlserver/sample.wsdl",
  "Wsd1Service": "Greeting",
  "Wsd1Port": "GreetingPort",
  "BasicAuthentication": "baFixed",
  "AuthUser": "bauser",
  "AuthPassword": "bapassword"
  "UsernameToken": "passworddigest",
}
```

Updating an XML/SOAP Mapping File

Command	Parameter	Description
sagcc update configuration data	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	<i>instanceid</i>	Required. The instanceid of an XML/SOAP mapping is prefixed with "XMM-" followed by a number.
	<i>-i file</i>	Optional. Specifies the file from where you want the input read.

Example

- To load the parameters of XML/SOAP mapping file with instance ID "XMM-1" in the installation with alias name "local" from file *mapping.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerXml-MyRpcServer XMM-1 -i mapping.json
```

See [Example 2](#) above for sample output file.

Displaying the EntireX Inventory

Listing all Inventory Components

The following table lists the parameters to include, when listing all EntireX instances, using the Command Central `list inventory` commands.

Command	Parameter	Description
sagcc list inventory components	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>component_id</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To list inventory components of instance EntireX in the installation with alias name "local":

```
sagcc list inventory components local EntireXCore*
```

A list of all EntireX RPC Server runtime components will be displayed.

Viewing the Runtime Status

The following table lists the parameters to include when displaying the state of an EntireX component, using the Command Central `get monitoring` commands.

Command	Parameter	Description
sagcc get monitoring state	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To display state information about the RPC Server for XML/SOAP:

```
sagcc get monitoring state local EntireXCore-RpcServerXml-MyRpcServer
```

Runtime status and runtime state will be displayed.

- Runtime *status* indicates whether a runtime component is running or not. Examples of a runtime status are ONLINE or STOPPED.
- Runtime *state* indicates the health of a runtime component by providing key performance indicators (KPIs) for the component. Each KPI provides information about the current use, marginal use, critical use and maximum use.

Starting an RPC Server Instance

The following table lists the parameters to include when starting an EntireX RPC Server for XML/SOAP, using the Command Central `exec lifecycle` commands.

Command	Parameter	Description
sagcc exec lifecycle start	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To start the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle start local EntireXCore-RpcServerXml-MyRpcServer
```

Information about the job - including the job ID - will be displayed.

Stopping an RPC Server Instance

The following table lists the parameters to include when stopping an EntireX RPC Server for XML/SOAP, using the Command Central `exec lifecycle` commands.

Command	Parameter	Description
sagcc exec lifecycle stop	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To stop the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle stop local EntireXCore-RpcServerXml-MyRpcServer
```

Information about the job - including the job ID - will be displayed.

Inspecting the Log Files

Here you can administer the log files of the RPC Server for XML/SOAP. The following table lists the parameters to include when displaying or modifying parameters of the RPC server, using the Command Central `list` commands.

- [List all RPC Server Log Files](#)
- [Getting Content from or Downloading RPC Server Log Files](#)

List all RPC Server Log Files

Command	Parameter	Description
sagcc list diagnostics logs	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To list the log files of RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc list diagnostics logs local EntireXCore-RpcServerXml-MyRpcServer
```

Getting Content from or Downloading RPC Server Log Files

Command	Parameter	Description
sagcc get diagnostics logs	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	full tail head	Optional. Shows full log file content, or only tail or head.
	export -o <i>file</i>	Optional. Creates a zip file of the logs.

Example 1

- To list the tail of the log file content in the current working directory:

```
sagcc get diagnostics logs local EntireXCore-RpcServerXml-MyRpcServer server.log  
tail
```

Example 2

- To create a zip file *myfile.zip* of the logs:

```
sagcc get diagnostics logs local EntireXCore-RpcServerXml-MyRpcServer export -o  
myfile.zip
```


Changing the Trace Level Temporarily

Here you can temporarily change the trace level of a running RPC server. The following table lists the parameters to include when displaying or modifying parameters of an EntireX component, using the Command Central `exec administration` command. The change is effective immediately; there is no need to restart the RPC server.



Note: If you want to set the trace level permanently, see [Trace](#) under *Configuring an RPC Server Instance*.

Displaying the Trace Level of a Running RPC Server

Command	Parameter	Description
sagcc exec administration	component	Required. Specifies that a component will be administered.
	node_alias	Required. Specifies the alias name of the installation in which the runtime component is installed.
	Trace	Required. Specifies what is to be administered.
	load tracelevel=?	Required. Get the trace level.
	-f xml json	Required. Specifies XML or JSON as output format.

Example 1

- To display the current trace level of the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" in JSON format on stdout:

```
sagcc exec administration component local EntireXCore-RpcServerXml-MyRpcServer
Trace load tracelevel=? -f json
```

Example 2

- To display the current trace level of the RPC Server for XML/SOAP "MyRpcServer" in the installation with alias name "local" in XML format on stdout:

```
sagcc exec administration component local EntireXCore-RpcServerXml-MyRpcServer
Trace load tracelevel=? -f xml
```

Updating the Trace Level of a Running RPC Server

Command	Parameter	Description
sagcc exec administration	component	Required. Specifies that a component will be administered.
	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".
	Trace	Required. Specifies what is to be administered.
	update tracelevel	Required. Update temporarily the trace level of a running RPC server.
	-f xml json	Required. Specifies XML or JSON as output format.

Example

- To change the current trace level of the running RPC Server with the name "MyRpcServer" in the installation with alias name "local":

```
sagcc exec administration component local EntireXCore-RpcServerXml-MyRpcServer
Trace update tracelevel=2 -f json
```

Deleting an RPC Server Instance

The following table lists the parameters to include when deleting an EntireX RPC Server instance, using the Command Central `delete instances` commands.

Command	Parameter	Description
sagcc delete instances	<i>node_alias</i>	Required. Specifies the alias name of the installation in which the runtime component is installed.
	<i>componentid</i>	Required. The component identifier. The prefix is "EntireXCore-RpcServerXml-".

Example

- To delete an instance of an EntireX RPC Server for XML/SOAP with the name "MyRpcServer" in the installation with alias name "local":

```
sagcc delete instances local EntireXCore-RpcServerXml-MyRpcServer
```

Information about the deletion job - including the job ID - is displayed.

5 Administering the RPC Server for XML/SOAP with Local

Scripts

▪ Customizing the RPC Server	66
▪ Configuring the RPC Server	68
▪ Using SOAP 1.2 with the RPC Server	72
▪ Locating and Calling the Target Server	72
▪ Using SSL/TLS with the RPC Server	78
▪ Starting the RPC Server	81
▪ Stopping the RPC Server	81
▪ Pinging the RPC Server	82
▪ Running an EntireX RPC Server as a Windows Service	82

The EntireX RPC Server for XML/SOAP allows RPC clients to communicate with target servers via HTTP(S). The RPC Server for XML/SOAP transforms RPC client calls into XML/SOAP calls. It works together with the XML/SOAP Wrapper.

This chapter describes how to administer the RPC Server for XML/SOAP with local scripts as in earlier versions of EntireX.

See also *Administering the RPC Server for XML/SOAP* with the Command Central [GUI](#) | [Command Line](#).

Customizing the RPC Server

The following elements are used to set up the RPC Server for XML/SOAP:

- [Start Script](#)
- [Properties File](#)
- [System Properties](#)
- [Configuration File](#)

Start Script

The start script for the RPC Server for XML/SOAP is called *jxmlrpcserver.bsh* (UNIX) or *jxmlrpcserver.bat* (Windows) and is provided in the *bin* folder of the installation directory. You may customize this file.

You can set the environment variable `JAVA_HOME` for the location of the Java interpreter. Set the classpath to `entirex.jar`, `wsstack-client.jar` and `saglic.jar`.

The RPC Server for XML/SOAP accepts two unnamed parameters, *broker_id* and *server_address*. Default values are `localhost:1971` and `RPC/SRV/CALLNAT`. Example:

```
java -Dentirex.license.location=<license.xml with path> ↵  
com.softwareag.entirex.xml.rt.XMLRPCServer broker_id server_address
```

Properties File

The default name of the properties file is `entirex.xmlrpcserver.properties`. The file is searched for in the directory of the [Start Script](#). It can be changed by assigning an arbitrary file name with a path to a property with the name `entirex.xmlrpcserver.propertyfile`.

The following is a sample properties file `entirex.xmlrpcserver.properties`:

```
# Example server configuration
#
# parameter for xml stream parser
entirex.sdk.xml.runtime.xmlparserfactory=com.ctc.wstx.stax.WstxInputFactory
# xmlruntime configuration file
entirex.sdk.xml.runtime.configurationfile=entirex.xmlrpcserver.configuration.xml
#
# Basic properties
entirex.server.brokerid=localhost
entirex.server.serveraddress=RPC/XMLSERVER/CALLNAT
entirex.server.userid=XMLRPCServer
```

It can define parser settings, for example JAXP parameters (optional if these parameters are already specified in your environment) and the location of the configuration file. A sample properties file is contained in subfolder *config* of the installation folder.

System Properties

Additionally, Java system properties are available to administer the RPC Server for XML/SOAP. These properties are independent of the administration possibilities listed above.

Java System Property	Description	Values	Default
<code>http.keepAlive</code>	Enable/disable HTTP persistence	true, false	true
<code>http.maxConnections</code>	Define the maximum number of HTTP connections to a host. Note: Requires <code>http.keepAlive=true</code>	Integer > 0	5
<code>http.maxTotalConnections</code>	Define the maximum total number of HTTP connections to a host. If <code>http.maxConnections</code> is greater than <code>http.maxTotalConnections</code> , this value is set to value of <code>http.maxConnections</code> . Note: Requires <code>http.keepAlive=true</code>	Integer > 0	20

Configuration File

The configuration contains a list of target servers defined in the [TargetServer](#) block, including XML mapping files (XMM) associated with them and may contain information about the broker if not already given as a property (command-line option, system property or property file).

The default name of the configuration file is `entirex.xmlrpcserver.configuration.xml`. The file is searched for in the directory of the [Start Script](#). It can be changed by assigning an arbitrary file name with a path to a property with the name `entirex.xml.runtime.configurationfile`.

The configuration file has the lowest priority. It is written in XML format. The document frame is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EntireX xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration" ←
version="10.8.0" >
  <XmlRuntime Version="1">
    <!-- information for RPC Server for XML/SOAP, see TargetServer Block -->
  </XmlRuntime>
</EntireX>
```

Configuring the RPC Server

The RPC Server for XML/SOAP uses the properties that start with “entirex.server” for configuring the RPC server side.

Alternatively to the properties, you can use the command-line options. These have a higher priority than the properties set as Java system properties, and these have higher priority than the properties in the configuration file.

Property Name	Command-line Option	Default	Explanation
entirex.server.brokerid	-broker	localhost	Broker ID.
entirex.server.codepage	-codepage		<p>The encoding configured for the Java virtual machine (JVM) is used to convert the Unicode (UTF-16) representation within Java to the encoding sent to or received from the broker by default. This encoding is also transferred as the codepage to the broker to tell the broker the encoding of the data. Changing the default encoding of the JVM has the side effect that the encoding for terminal and file IO is affected too. This may be undesired.</p> <p>With the codepage parameter you can override the encoding without the need to change the default encoding of the JVM. The codepage must be supported by your JVM. For a list of valid encodings, see <i>Supported Encodings</i> in your Java documentation.</p> <p>Note: See your JVM documentation for how to change the default encoding of the JVM. On some JVM implementations, it can be</p>

Property Name	Command-line Option	Default	Explanation																
			<p>changed with the <code>file.encoding</code> property. On some UNIX implementations, it can be changed with the <code>LANG</code> environment variable.</p> <p>Enable character conversion in the broker by setting the service-specific attribute <code>CONVERSION</code> to "SAGTRPC". See also <i>Configuring ICU Conversion</i> under <i>Configuring Broker for Internationalization</i> in the platform-specific Administration documentation. More information can be found under <i>Internationalization with EntireX</i>.</p> <p>See also Character Encoding for Outgoing XML Document.</p>																
<code>entirex.server.compresslevel</code>	<code>-compresslevel</code>	0 (no compression)	<p>Enter the text or the numeric value:</p> <table border="0"> <tr> <td><code>BEST_COMPRESSION</code></td> <td>9</td> </tr> <tr> <td><code>BEST_SPEED</code></td> <td>1</td> </tr> <tr> <td><code>DEFAULT_COMPRESSION</code></td> <td>-1</td> </tr> <tr> <td></td> <td>(mapped to 6)</td> </tr> <tr> <td><code>DEFLATED</code></td> <td>8</td> </tr> <tr> <td><code>NO_COMPRESSION</code></td> <td>0</td> </tr> <tr> <td><code>N</code></td> <td>0</td> </tr> <tr> <td><code>Y</code></td> <td>8</td> </tr> </table>	<code>BEST_COMPRESSION</code>	9	<code>BEST_SPEED</code>	1	<code>DEFAULT_COMPRESSION</code>	-1		(mapped to 6)	<code>DEFLATED</code>	8	<code>NO_COMPRESSION</code>	0	<code>N</code>	0	<code>Y</code>	8
<code>BEST_COMPRESSION</code>	9																		
<code>BEST_SPEED</code>	1																		
<code>DEFAULT_COMPRESSION</code>	-1																		
	(mapped to 6)																		
<code>DEFLATED</code>	8																		
<code>NO_COMPRESSION</code>	0																		
<code>N</code>	0																		
<code>Y</code>	8																		
<code>entirex.server.development.relativepaths</code>		false	The file locations of deployed XMM and WSDL files are written as relative paths in configuration file of the RPC Server for XML/SOAP.																
<code>entirex.server.fixedservers</code>		no	<p>NO The number of worker threads balances between what is specified in <code>entirex.server.minservers</code> and what is specified in <code>entirex.server.maxservers</code>. This is done by a so-called attach thread. At startup, the number of worker threads is the number specified in <code>entirex.server.minservers</code>. A new worker thread starts if the broker has more requests than there are worker threads waiting. If more than the number specified in <code>entirex.server</code>.</p>																

Property Name	Command-line Option	Default	Explanation
			<p>minservers are waiting for requests, a worker thread stops if its receive call times out. The timeout period is configured with <code>entirex.server.waitserver</code>.</p> <p>See worker model DYNAMIC.</p> <p>YES The number of worker threads specified in <code>entirex.server.minservers</code> is started and the server can process this number of parallel requests. See worker model FIXED.</p>
<code>entirex.server.ignoreSOAPActionNamespace</code>		false	<p>Possible values:</p> <p>true Only the name part of SOAPAction is used, the namespace is ignored.</p> <p>false The SOAPAction value is used as defined.</p> <p>Note: If a WSDL file is configured for this method, the property will be ignored and SOAPAction value defined in WSDL is used.</p>
	-help		Display usage of the command-line parameters.
<code>entirex.server.logfile</code>	-logfile		Path and name of the log file. Environment variables in the name are resolved only if used as command-line option.
<code>entirex.server.minservers</code>		1	Minimum number of server threads.
<code>entirex.server.maxservers</code>		32	Maximum number of server threads.
<code>entirex.server.name</code>			Server name.
<code>entirex.server.password</code>	-password	yes	<p>The password for secured access to the broker. The password is encrypted and written to the property <code>entirex.server.password.e</code>.</p> <ul style="list-style-type: none"> ■ To change the password, set the new password in the properties file. ■ To disable password encryption, set <code>entirex.server.passwordencrypt=no</code>. Default=yes.

Property Name	Command-line Option	Default	Explanation
entirex.sdk.xml.runtime.propertyfile	-propertyfile	entirex.xmlrpcserver.properties	Name of property file.
entirex.sdk.xml.runtime.configurationfile	-configurationfile	entirex.xmlrpcserver.configuration.xml	Location and name of configuration file.
entirex.server.restartcycles	-restartcycles	15	Number of restart attempts if the Broker is not available. This can be used to keep the RPC Server for XML/SOAP running while the Broker is down for a short time.
entirex.server.security	-security	no	Valid values: no yes auto name of BrokerSecurity object.
entirex.server.serveraddress	-server	RPC/SRV1/CALLNAT	Server address.
entirex.server.serverlog	-serverlog		Name of the file where start and stop of the worker threads is logged. Used by the Windows RPC Service.
entirex.server.userid	-user	XMLRPCServer	The user ID for access to the broker.
entirex.server.waitattach		600S	Wait timeout for the attach server thread.
entirex.server.waitserver		300S	Wait timeout for the worker threads.
entirex.timeout		20	TCP/IP transport timeout.
entirex.trace	-trace	0	Trace level. 0 No tracing, default. 1 Trace all broker calls and other major actions. 2 Dump the send and receive buffer. 3 Dump the buffers sent to the broker and received from the broker.
entirex.sdk.xml.runtime.xmlparserfactory	-jaxp.saxparserfactory	com.ctc.wstx.stax.WstxInputFactory	Location and name of stream parser factory class.
entirex.sdk.xml.runtime.defaultFaultDocumentFormat	soap	soap	Define the protocol used for fault document generation if no fault document is defined. Values: soap xml.

Using SOAP 1.2 with the RPC Server

Generated Mapping (XMM)

Generate a new XML mapping file with SOAP mapping in the *XML Mapping Editor*, or use an existing one.



Note: Do not change the namespace for SOAP. This must be the namespace for SOAP version 1.1.

Configure XMM

To use SOAP 1.2 for an XMM file, add the attribute `soapVersion` to section `xmms` in element `ex-xmm` in the configuration file. See `soapVersion` under *TargetServer Block* below for details.

Locating and Calling the Target Server

The IDL library and IDL program names that come from the RPC client are used to locate the target server. See `library-definition` and `program-definition` under *Software AG IDL Grammar* in the IDL Editor documentation. This two-level concept (library and program) is mapped to a target servers using an XML mapping file (XMM) together with connection information. Both connection information and XML mapping files are defined in the target server block of the *Configuration File*.

Mapping Software AG IDL files to XML/SOAP is described under *XML Structures and IDL-XML Mapping* in the XML/SOAP Wrapper documentation. This section describes the following elements to define servers:

- [TargetServer Block](#)
- [HTTP Basic Authentication](#)
- [UsernameToken Security](#)
- [WS-* Features \(e.g. WS-Policy\)](#)
- [User Exit to Set Additional HTTP Headers](#)

- [Character Encoding for Outgoing XML Document](#)

TargetServer Block

The section `<TargetServer>` specifies a target server with mandatory HTTP(S) Web service address in the name attribute.



Tip: If you are using a target located outside the firewall, set the following Java properties:

- `http.proxyHost`
- `http.proxyPort`
- `http.nonProxyHosts`
- `http.proxyUser`
- `http.proxyPassword`

Configuring SSL/TLS to the target server is described under [Using SSL/TLS with the RPC Server](#).

The block contains section `<xmms>` with EntireX XML mapping files (XMM files). Each `<TargetServer>` entry can have a list of XMMs for its server.



Caution: It is not allowed to use one XMM in more than one `TargetServer` entry inside one configuration file.

Within the `<TargetServer>` tag you can specify basic authentication with a fixed user/password:

Attribute	Req/ Opt	Description
<code>basicAuthentication</code>	O	<p><code>true</code> Activate the basic authentication. If attributes <code>user</code> and <code>password</code> are set, these credentials are used for basic authentication. Otherwise the current credentials of the calling RPC client are used.</p> <ul style="list-style-type: none"> ■ For how to send the RPC user ID/password pair from an RPC client, see <i>Using the Broker and RPC User ID/Password</i> (.NET Wrapper Java Wrapper C Wrapper PL/I Wrapper DCOM Wrapper Web Services Wrapper IDL Tester Listener for XML/SOAP Listener for IBM MQ). ■ For the COBOL Wrapper, refer to <i>Using Broker Logon and Logoff</i> and <i>Using RPC Authentication (Natural Security, Impersonation, Integration Server)</i>.

Attribute	Req/ Opt	Description
		<ul style="list-style-type: none"> ■ For non-RPC clients, see <i>Using the Broker and RPC User ID/Password</i> under <i>EntireX XML Tester</i> in the XML/SOAP Wrapper documentation. <p>false Deactivate basic authentication. All other parameters in this table are ignored.</p>
user	O	Name of default user for basic authentication.
password	O	Password of default user for basic authentication.
password-encryption	O	Specifies how the password is encrypted. Possible values: plainText Default. base64 encrypt The RPC Server for XML/SOAP encrypts the password and sets this value.
httpConnectionTimeout	R	HTTP connection timeout in seconds.

The section <xmms> contains the optional attributes for SOAP mapping.

Attribute	Description
soapVersion	Specifies a SOAP version: 1.1 (default) or SOAP 1.2.
wSDL	The location of WSDL file, using a WSDL file the target address is retrieved from WSDL file.
service	The service name in WSDL file.
port	The port name in WSDL file.
repository	The repository directory used for WS-* features. See Software AG Web Services Stack client repository.
usernameToken	Valid values: PasswordText PasswordDigest. Prerequisites: Attribute repository must be defined and module rampart must be engaged. See also UsernameToken Security .

Sample configuration file:

```

<?xml version="1.0" encoding="UTF-8" ?>
<EntireX
  xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration"
  version="10.8">
  <XmlRuntime Version="1">

    <TargetServer name="http://localhost:1973/MyService">
      <xmms>
        <exx-xmm name="c:\mydir\xmmfiles\XmmExample.xmm"
          soapVersion="1.1"
          wsdl="c:/mywsdl.wsdl" service="myservice"
          port="myserviceSOAP11Port" repository="c:\myrepository"\>
        </xmms>
      </TargetServer>
    </XmlRuntime>
  </EntireX>

```

HTTP Basic Authentication

Basic authentication is used for a target server if the attribute `basicAuthentication` is defined in the `TargetServer` block. Basic authentication is used for all calls associated with the defined XMM files for the `<TargetServer>`.

Basic authentication can be used with fixed credentials or credentials set from the RPC client application:

- If `<TargetServer>` contains attributes `user` and `password`, these settings are used for basic authentication.
- Otherwise the RPC client application must provide the credentials:
 - For how to send the RPC user ID/password pair from an RPC client, see *Using the Broker and RPC User ID/Password* (.NET Wrapper | Java Wrapper | C Wrapper | PL/I Wrapper | DCOM Wrapper | Web Services Wrapper | IDL Tester | Listener for XML/SOAP | Listener for IBM MQ).
 - For the COBOL Wrapper, refer to *Using Broker Logon and Logoff* and *Using RPC Authentication (Natural Security, Impersonation, Integration Server)*.
 - For non-RPC clients, see *Using the Broker and RPC User ID/Password* under *EntireX XML Tester* in the XML/SOAP Wrapper documentation.

See also *Generating a Web Service with HTTP Basic Authentication and UsernameToken Authentication for EntireX Authentication* under *Using the EntireX Web Services Wrapper*.

UsernameToken Security

UsernameToken security is used for a target server if attribute `usernameToken` is defined in `<xmms>`. Two kinds of UsernameToken are supported:

- PasswordText
- PasswordDigest

The repository must be defined in section `<xmms>`, for example:

```
<exx-xmm name="AService.xmm" soapVersion="1.1"
repository="myrepository" usernameToken="PasswordText" />
```

The repository must contain module `rampart`. In the configuration file (`axis2.xml`) the `rampart` module must be engaged (`<module ref="rampart"/>`) and the phase `PreSecurity` can be empty (`<phase name="PreSecurity" />`).

Natural logon must be set in the RPC client application. Additionally the RPC client application should set RPC user ID and RPC password.

See also *Generating a Web Service with HTTP Basic Authentication and UsernameToken Authentication for EntireX Authentication* under *Using the EntireX Web Services Wrapper*.

WS-* Features (e.g. WS-Policy)

To use WS-* features such as WS-Policy you need to define the *repository location* (used by underlying Software AG Web Services Stack) and the WSDL file location, including *service name* and *port name* in the configuration file of the RPC Server for XML/SOAP. A sample repository is provided in the installation (`<installation_home>/WS-Stack`).

For specifying features such as WS-Policy, see configuration of Software AG Web Services Stack.

User Exit to Set Additional HTTP Headers

The following user exit allows you to add HTTP headers before the call, or - in case of HTTP error 401 (authentication) - to add an HTTP header to optional retry call, for example to support SPNEGO (Simple and Protected GSSAPI Negotiation Mechanism).

- [Writing the User Exit](#)
- [Adapting the Configuration of the RPC Server for XML/SOAP](#)
- [Adapting the Start Script](#)

- Client Application

Writing the User Exit

Implement the interface `com.softwareag.entirex.xml.rt.IHttpTransportUserExit` with the two methods `addHttpHeaders` and `retryOnAuthenticationError`. See also the sample implementation, including the Javadoc. Sample user exit:

```
package sample;

import java.util.Properties;

import com.softwareag.entirex.xml.rt.IHttpTransportUserExit;

/**
 * implementation of user exit must be thread-safe
 */
public class SampleHttpTransportUserExit implements IHttpTransportUserExit {

    private String getToken(String user, String endpoint) {
        // add your implementation here
        return "";
    }

    /**
     * Method to add HTTP header.
     * Method is called before first call and
     * if an HTTP error 401 occurred and retryOnAuthenticationError() is true
     * @param settings properties containing user and endpoint
     * @param httpHeaders
     */
    @Override
    public void addHttpHeaders(Properties settings, Properties httpHeaders) {
        String user = settings.getProperty("user");
        String endpoint = settings.getProperty("endpoint");

        httpHeaders.setProperty("Authorization", "Negotiate " + getToken(user, endpoint));
    }

    /**
     * @return should make a retry call on authentication error (401)
     */
    @Override
    public boolean retryOnAuthenticationError() {
        return true;
    }
}
```

Adapting the Configuration of the RPC Server for XML/SOAP

On element `TargetServer`, add the attribute `httpTransportUserExit` with the qualified name of the user exit in the configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<EntireX version="10.8" ↵
xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration">
  <XmlRuntime Version="1">
    <TargetServer name="http://localhost:8080/wsstack/services/exampleSOAP"
      httpConnectionTimeout="60"
      httpTransportUserExit="sample.SampleHttpTransportUserExit"
    >
  </TargetServer>
  <xmms>
    <exx-xmm name="C:/example.xmm" soapVersion="1.1" />
  </xmms>
</XmlRuntime>
</EntireX>
```

Adapting the Start Script

Add the user exit to `classpath` in the start script.

Client Application

To set the credentials - this means that the user ID is available in user exit on the client side - the Natural logon must be enabled. User-specific credentials can be overwritten by setting the RPC user ID and RPC password in the client application.

Character Encoding for Outgoing XML Document

The encoding for the outgoing XML document is determined by the XML Default Encoding defined in the XML mapping file. See *Defining the XML Encoding* in the XML Mapping Editor documentation.

Using SSL/TLS with the RPC Server

RPC servers can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term "SSL" in this section refers to both SSL and TLS. RPC-based servers are always SSL clients. The SSL server can be either the EntireX Broker or Broker SSL Agent. For an introduction see *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.

To use SSL with RPC Server for XML/SOAP, you need to configure two sides: the RPC server side and the target server side.

➤ To configure SSL/TLS with the RPC server side

- 1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.
- 2 Set up the RPC Server for XML/SOAP for an SSL connection.

Use the *URL-style Broker ID* with protocol `ssl://` for the Broker ID. If no port number is specified, port 1958 is used as default. Example:

```
ssl://localhost:22101?trust_store=C:\SoftwareAG\EntireX\etc\ExxCACert.jks&verify_server=no
```

If the SSL client checks the validity of the SSL server only, this is known as *one-way SSL*. The mandatory `trust_store` parameter specifies the file name of a keystore that must contain the list of trusted certificate authorities for the certificate of the SSL server. By default a check is made that the certificate of the SSL server is issued for the hostname specified in the Broker ID. The common name of the subject entry in the server's certificate is checked against the hostname. If they do not match, the connection will be refused. You can disable this check with SSL parameter `verify_server=no`.

If the SSL server additionally checks the identity of the SSL client, this is known as *two-way SSL*. In this case the SSL server requests a client certificate (the parameter `verify_client=yes` is defined in the configuration of the SSL server). Two additional SSL parameters must be specified on the SSL client side: `key_store` and `key_passwd`. This keystore must contain the private key of the SSL client. The password that protects the private key is specified with `key_passwd`.

The ampersand (&) character cannot appear in the password.

SSL parameters are separated by ampersand (&). See also *SSL/TLS Parameters for SSL Clients*.

- 3 Make sure the SSL server to which the RPC Server for XML/SOAP connects is prepared for SSL connections as well. The SSL server can be EntireX Broker or Broker SSL Agent. See:
 - *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
 - *Broker SSL Agent* in the platform-specific Administration documentation

➤ To configure SSL/TLS to a target server

- 1 Using HTTPS to the target server requires setting Java SSL/TLS properties and changing the protocol of the target server's Web service in the *Configuration File*.

If the web service has to be called via HTTPS (SSL/TLS), the SSL client (here the RPC Server for XML/SOAP) needs the correct certificate for the web service in the truststore to be able to

communicate via SSL. The certificate can either be stored in the default truststore of the JVM or in the truststore specified with the following Java property:

```
-Djavax.net.ssl.trustStore=path_to_used_truststore
```

The SSL parameters must be included in quotes, for example

```
set SSL="-Djavax.net.ssl.trustStore=C:\myTrustStore.jks"
```



Note: This is only an example. You must provide a truststore that matches your environment.

The truststore keeps the trusted certificate of the web service host or the certificate of its signing (issuing) certificate authority. In the event of an SSL error, you can use the Java property `-Djavax.net.debug=all` to get more information. Add the SSL parameter to the start script of the RPC Server for XML/SOAP and ensure it is passed to the start or Java. Example:

```
...
set SSL="-Djavax.net.ssl.trustStore=C:\myTrustStore.jks -Djavax.net.debug=all"
...
"%JAVA_HOME_BIN%java" %PROXY% %SSL% -classpath "%CLASSPATH%" ←
com.softwareag.entirex.xml.rt.XMLRPCServer -p "%EXXPROP%" -c "%EXXCONF%" %*
```

There are additional Java properties, which are usually not needed. These are described in the Java documentation.

- 2 Optional. If you are using an HTTPS target server's Web service address located outside the firewall, set the following Java properties:
 - `https.proxyHost`
 - `https.proxyPort`
 - `http.nonProxyHosts`
 - `https.proxyUser`
 - `https.proxyPassword`
- 3 Change the protocol of the target server's Web service address from `http` to `https` in the configuration file. Specify the fully qualified host name as `TargetServer`. The host name has to match the CN (Common Name) item of the host certificate. See also [Configuration File](#). Example:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<EntireX
xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration" ↵
version="10.8"
>
  <XmlRuntime Version="1">
    <TargetServer name="https://targethost:8080/entirex/xmlrt">
      <xmms>
        <exx-xmm name="yourFile1.xmm" />
        <exx-xmm name="yourFile2.xmm" />
      </xmms>
    </TargetServer>
  </XmlRuntime>
</EntireX>
```

Starting the RPC Server

➤ To start the RPC Server for XML/SOAP

- Use the [Start Script](#).

Or:

At the command prompt, enter:

```
java com.softwareag.entirex.aci.XMLRPCServer broker_id server_address
```

You can pass command-line options and customize your environment as described under [Start Script](#).

Stopping the RPC Server

➤ To stop the RPC Server for XML/SOAP

- Use the command `stopService`. See *Stop Running Services* in Command Central's Command-line Interface.

Or:

Stop the service using Command Central's Graphical User Interface. See *Stopping a Service*.

Or:

Use the command-line utility `etbcmd`. See `etbcmd` under *Broker Command-line Utilities* in the platform-specific Administration documentation.

Or:

Use `CTRL-C` in the session where you started the RPC server instance.

Or:

Under UNIX, enter command `kill -process-id`.

Pinging the RPC Server

➤ To ping the RPC Server for XML/SOAP

- Enter the following command:

```
java -classpath "$EXXDIR/classes/entirex.jar" ↵  
com.softwareag.entirex.rpcping.RPCServerPing -p <admin_port>
```

where `admin_port` is the number of the administration port.

The `ping` command returns "0" if the server is reachable, and "1" if the server cannot be accessed.



Note: This command is particularly useful in a high availability cluster context. See *Setting up your Environment for High Availability with Container Orchestration* in the High Availability documentation.

Running an EntireX RPC Server as a Windows Service

For general information see *Running an EntireX RPC Server as a Windows Service* in the Windows Administration documentation.

➤ To run the RPC Server for XML/SOAP as a Windows Service

- 1 Customize the *Start Script* according to your system installation.



Note: The script file must pass external parameters to the RPC server:

```
java com.softwareag.entirex.xml.rt.XMLRPCServer %*
```

See also [Starting the RPC Server](#).

- 2 Test your RPC server to see whether it will start if you run your script file.
- 3 Use the *EntireX RPC Service Tool* and install the `RPCService` with some meaningful extension, for example `MyServer`. If your *Start Script* is `jxmlrpcserver.bat`, the command will be

```
RPCService -install -ext MyServer ↵  
-script install_path\EntireX\bin\jxmlrpcserver.bat
```

The log file will be called `RPCservice_MyServer.log`.

- 4 In **Windows Services** menu (**Control Panel** > **Administrative Tools** > **Services**) select the service: `Software AG EntireX RPC Service [MyServer]` and change the property `Startup Type` from "Manual" to "Automatic".

6 Java API

- Properties File 86
- Configuration File 86
- Implementation of the Java API for the RPC Server for XML/SOAP 88
- Start Script 89

The Java API allows you to direct RPC client calls to Java objects instead of to target servers via HTTP(s). A special configuration is required to call Java objects; see [Configuration File](#). The [Start Script](#) is also different. From the RPC client point of view, usage is the same.

Properties File

The Java API uses the same [Properties File](#) as the RPC Server for XML/SOAP.

Configuration File

The Java API uses the same [Configuration File](#) as the RPC Server for XML/SOAP.

The services (programs) directed to the Java interface of the RPC Server for XML/SOAP have to use a special keyword `xmlrpcServerClass` as the value of the `name` attribute in the [TargetServer](#) block. A mixture of target servers using HTTP(S) and Java objects is possible.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EntireX
xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration"
version="10.8"
>
<XmlRuntime Version="1">
<BrokerInfo>
<BrokerId>localhost:1971</BrokerId>
<ServerAddress>RPC/SRV1/CALLNAT</ServerAddress>
</BrokerInfo>
<TargetServer name="xmlrpcServerClass">
<xmms>
<exx-xmm name="java-service1.xmm" />
<exx-xmm name="java-service2.xmm" />
<exx-xmm name="java-service3.xmm" />
</xmms>
</TargetServer>
<TargetServer name="http://myWebService">
<xmms>
<exx-xmm name="http-service1.xmm" />
<exx-xmm name="http-service2.xmm" />
</xmms>
</TargetServer>
</XmlRuntime>
</EntireX>
```

Implementation of the Java API for the RPC Server for XML/SOAP

The Java API requires a user-written Java class initializing the RPC Server for XML/SOAP and implementing the XMLRPCServerInterface.

Example:

```
import java.util.Properties;
import com.softwareag.entirex.xml.rt.XMLRPCServerInterface;
import com.softwareag.entirex.xml.rt.XMLRPCServer;
public class MyXMLRPCServer implements XMLRPCServerInterface
{
    public MyXMLRPCServer ()
    {
        XMLRPCServer xmlRpcServer = new XMLRPCServer();
        // register your implementation of XMLRPCServerInterface
        xmlRpcServer.registerXMLRPCServerClass ((XMLRPCServerInterface) this);
        // start RPC Server for XML/SOAP with arguments (same as command line)
        xmlRpcServer.start(new String[0]);
    }

    // mandatory method invoke (from XMLRPCServerInterface)
    // - thread synchronization must be done by application if required
    // - properties object contains property "charset" (as used in xml-declaration)
    // and property "java.charset" - the corresponding Java codepage
    // - Exception thrown from this method is mapped to error class 2000 and error ←
    number 200,
    // with exception information in errortext

    public byte[] invoke(byte[] requestDocument, Properties properties)
        throws Exception
    {
        byte[] response = null;
        // TODO <insert application code here>
        return response;
    }

    public static void main(String[] args)
    {
        MyXMLRPCServer myServer = new MyXMLRPCServer ();
    }
}
```

Start Script

The RPC Server for XML/SOAP with Java API must be started by implementing XMLRPCServerInterface as in this example:

```
java -classpath "%PARSER%;%CLASSPATH%" MyXMLRPCServer
```


7 Running the RPC Server for XML/SOAP in the Software AG

Runtime

▪ Introduction	92
▪ Configuration	93
▪ Deactivating an RPC Server for XML/SOAP Permanently	93
▪ Starting and Stopping the RPC Server for XML/SOAP using JMX (Java Management Extensions)	93
▪ Starting and Stopping the Software AG Runtime under UNIX	94
▪ Starting and Stopping the Software AG Runtime under Windows	94

The RPC Server for XML/SOAP allows RPC clients to communicate with target servers via HTTP(S). The RPC Server for XML/SOAP transforms RPC client calls into XML/SOAP calls. It enables standard RPC clients to communicate with XM/SOAP servers. It works together with the XML/SOAP Wrapper.

The RPC Server for XML/SOAP can run in the Software AG Runtime. This chapter covers the following topics:

See also *RPC Server for XML/SOAP in the Software AG Runtime* under *Frequently Asked Questions (FAQ) and Troubleshooting* in the XML/SOAP Wrapper documentation.

Introduction

The Software AG Common Platform is a Java Runtime environment based on the OSGi framework. It provides a standard platform on which to run Software AG products and the enterprise applications you develop around those products. The Software AG Common Platform provides common infrastructure for user authentication, event handling, and the execution of Web applications. Infrastructure components that the Software AG Common Platform provide include Software AG Security Infrastructure, Software AG Web Server based on Apache Tomcat, and Web Services Stack.

The Software AG Runtime is an installable instance of the Software AG Common Platform that functions as a stand-alone Tomcat server and a container for Web applications. EntireX uses the Software AG Runtime to host the EntireX Listener for XML/SOAP and RPC Server for XML/SOAP.

The Software AG Web Server based on Apache Tomcat is one of the basic infrastructure components provided by the Software AG Common Platform. It provides HTTP/HTTPS services, a JSP engine, and a servlet container. Unlike a typical Tomcat implementation, the Software AG Web Server is OSGi-based and supports both .WAR-based and .WAB-based web applications.

During startup, the Software AG Web Server (service name: Software AG Runtime), including the EntireX bundle, looks in the EntireX profile for file `<Installation home>/EntireX/etc/EXX/workspace/entirex.servers.properties`. This file defines an RPC Server for XML/SOAP as within `entirex.xmlrpcserver.properties` and `entirex.xmlrpcserver.configuration.xml` located in the EntireX installation in subdirectory `config` by default.

After installation, the default RPC Server for XML/SOAP is not started automatically. To enable autostart, change the configuration file (set property `start` to "yes") and restart the Software AG Runtime.

Configuration

The file *entirex.servers.properties* defines the servers to be started. It is only read during startup of the Software AG Runtime. Set the following properties for each defined server:

Property Name	Description
<code>server.n.start</code>	Enable autostart with "yes" or disable autostart with "no" (default).
<code>server.n.propertiesFile</code>	Path to properties file (Java notation).
<code>server.n.configurationFile</code>	Path to configuration file (Java notation).

where *n* is a number identifying the server

Example of *entirex.servers.properties*:

```
server.1.start=yes
server.1.propertiesFile=c:/SoftwareAG/EntireX/config/entirex.xmlrpcserver.properties
server.1.configurationFile=c:/SoftwareAG/EntireX/config/entirex.xmlrpcserver.configuration.xml
server.2.start=no
server.2.propertiesFile=c:/SoftwareAG/EntireX/config/entirex.myxmlrpcserver.properties
server.2.configurationFile=c:/SoftwareAG/EntireX/config/entirex.myxmlrpcserver.configuration.xml
```

Deactivating an RPC Server for XML/SOAP Permanently

To deactivate an RPC Server for XML/SOAP, change its `start` property in the configuration file to "no".

Starting and Stopping the RPC Server for XML/SOAP using JMX (Java Management Extensions)

To start and stop an RPC Server for XML/SOAP, open a JMX tool, for example the Java Monitoring and Management Console (`jconsole`), located in the Java *bin* directory (sample path: `C:\Software-AG\jvm\w64_160\bin\jconsole.exe`). The tool should be connected to the Software AG Runtime JMX port remotely. The default number of this port is 8044 and is defined in `<Installation home>/profiles/CTP/configuration/config.ini`.

Switch to tab MBeans and select item `com.softwareag.entirex.runtime.rpcserver`. The following operations are available:

Operation	Description
startServer	To start a registered and non-running RPC Server for XML/SOAP. The parameter is the service name (e.g. RPC/XMLSERVER/CALLNAT).
stopServer	To stop a running RPC Server for XML/SOAP. The parameter is the service name (e.g. RPC/XMLSERVER/CALLNAT).
registeredServer	Returns the list of service names of all configured RPC Server for XML/SOAPs.
runningServer	Returns the list of service names of running configured RPC Server for XML/SOAPs.
nonRunningServer	Returns the list of service names of non-running configured RPC Server for XML/SOAPs.

Starting and Stopping the Software AG Runtime under UNIX

Under UNIX, the Software AG Runtime can be stopped and started using the following scripts:

```
suite_install_dir/profiles/CTP/bin/sagctpnn.sh stop  
suite_install_dir/profiles/CTP/bin/sagctpnn.sh start
```

where *nn* represents the product version number.

Starting and Stopping the Software AG Runtime under Windows

Under Windows, the Software AG Runtime runs as a service and can be started and stopped using the Windows Administrative tools.

System Service

The relevant CTP profile has a Windows service named **Software AG Runtime**. You can see it in the Service dialog from the Windows Control Center. You can start/stop the service from this dialog.

Batch Scripts

The directory '<Installation Home>\ profile\CTP\bin' contains the following scripts to start and stop the Software AG Runtime:

Script	Description
start_runtime.bat	batch script for starting a profile's runtime. If it is an installed service for the profile, it will start the service by default.
stop_runtime.bat	batch script for stopping a profile's runtime. If it is an installed service for the profile and this service has been started, this script will stop the service.

8

Building an EntireX RPC Server for XML/SOAP Docker Image

- Prerequisites 98
- Building and Running the RPC Server for XML/SOAP Image 98
- Verifying the Build 102
- Healthcheck for RPC Server for XML/SOAP 103

Prerequisites

- Operating system Linux
- Docker installation 1.13.1 or compatible
- Software AG EntireX installation containing the package EntireX > Core Files

Building and Running the RPC Server for XML/SOAP Image

The scripts provided with EntireX support the following three methods of building a Docker image and running the Docker container.

- [Configuring with Modified Dockerfile](#)
- [Configuring during Image Start, using Default File Names](#)
- [Configuring during Image Start, using Custom File Names](#)

Configuring with Modified Dockerfile

» To copy license and configuration files into Docker container

- 1 Set your working directory to `<install_dir>/EntireX/docker/XMLSoapRpcServer`.
- 2 Create the TAR file containing all the necessary files with the following command:

```
./CreateEntireXMLSoapRpcServerTar.sh
```

- 3 Provide your installation and configuration files into the current working directory, for example:

- *myLicense.xml*
- *myConfiguration*
- *myWebServerConfiguration*
- *myFirstMapping.xmm*
- *mySecondMapping.xmm*



Note: All files are required if you are using this method.

- 4 Update the Dockerfile, for example:

```

# Possibility to add a valid license file already to the image instead of ←
providing it during start up
# e.g.:
ADD myLicense.xml $EXXDIR/config/license.xml

# Possibility to add the configuration file already to the image instead of ←
providing it during start up
# e.g.:
ADD myConfiguration /configs/entirex.xmlrpcserver.properties

# Possibility to add web server configuration file already to the image instead ←
of providing it during start up
# e.g.:
ADD myWebServerConfiguration /configs/entirex.xmlrpcserver.configuration.xml

# Possibility to add custom mapping file already to the image instead of ←
providing it during start up
# e.g.:
ADD *.xmm /configs

```

5 Build the server image:

```
docker build -t exx_xml_soap_rpc_server_image_1 .
```

In this case the Docker `build` command copies the configuration into the image.

6 Start the container:

```
docker run -d -e ACCEPT_EULA=Y --name exx_xml_soap_rpc_server_container_1 ←
exx_xml_soap_rpc_server_image_1
```

■ Advantages

The complete configuration is in the image. For troubleshooting, Software AG Support will require only the image and the command you entered.

■ Disadvantage

If the configuration changes, you will have to build a new image before you rerun the container.

Configuring during Image Start, using Default File Names

➤ To copy license, configuration and server implementation files into container, using default file names

- 1 Set your working directory to `<install_dir>/EntireX/docker/XmlSoapRpcServer`.
- 2 Create the TAR file containing all the necessary files with the following command:

```
./CreateEntireXXmlSoapRpcServerTar.sh
```

3 Build the server image:

```
docker build -t exx_xml_soap_rpc_server_image_2 .
```

4 Provide your license, configuration and server implementation files with the default file names, for example:

- *<my-license-dir>/license.xml*
- *<my-config-dir>/entirex.xmlrpcserver.properties*
- *<my-config-dir>/entirex.xmlrpcserver.configuration.xml*
- *<my-config-dir>/<myMapping>.xmm*



Note: All files are required if you are using this method.

In this case the license and configuration files are mounted during startup.

5 Start the container:

```
docker run -d -e ACCEPT_EULA=Y
            -v <my-license-dir>:/licenses
            -v <my-config-dir>:/configs
            ↵
--name exx_xml_soap_rpc_server_container_2 exx_xml_soap_rpc_server_image_2
```

■ **Advantages**

If the configuration changes, you do not need to rebuild the image; you only need to rerun the container.

■ **Disadvantage**

The configuration, license and data files are mounted to the container. For troubleshooting, Software AG Support will require the image, configuration, license, data files and the command you entered.

Configuring during Image Start, using Custom File Names

➤ To copy license, configuration and server implementation files into container, using custom file names

(Customer service implementation JAR file are provided in directory mounted to */data*.)

- 1 Set your working directory to *<install_dir>/EntireX/docker/XmlSoapRpcServer*.
- 2 Create the TAR file containing all the necessary files with the following command:


```
./CreateEntireXXmlSoapRpcServerTar.sh
```

3 Build the server image:

```
docker build -t exx_xml_soap_rpc_server_image_3 .
```

4 Provide your configuration files with your own file names, for example:

- *<my-license-dir>/myLicense*
- *<my-config-dir>/myWebServerConfiguration*
- *<my-config-dir>/myConfiguration*
- *<my-config-dir>/myMapping.xmm*



Note: All files are required if you are using this method.

In this case the license and configuration files are mounted during startup.

5 Start the container:

```
docker run -d
    -e ACCEPT_EULA=Y
    -e "EXX_LICENSE_KEY=myLicense.xml"
    -e "EXX_XML_SERVER_CONFIGURATION=myConfiguration"
    -e "EXX_XML_SERVER_MAPPING=myWebServerConfiguration"
    -v <my-license-dir>:/licenses
    -v <my-configuration-dir>:/configs
    ↵
--name exx_xml_soap_rpc_server_container_3 exx_xml_soap_rpc_server_image_3
```

■ Advantages

If the configuration changes, you do not need to rebuild the image; you only need to rerun the container. You can choose your own file names.

■ Disadvantage

The configuration, license and data files are mounted to the container. For troubleshooting, Software AG Support will require the image, configuration, license, data files and the command you entered.

Verifying the Build

> To verify the build

- 1 Show the image with command

```
docker images
```

- 2 Start the docker image to be verified as described above, for example:

```
docker run -d -e ACCEPT_EULA=Y ↵  
--name exx_xml_soap_rpc_server_container_1 exx_xml_soap_rpc_server_image_1
```

- 3 Show the log:

```
docker logs -f exx_xml_soap_rpc_server_container_1
```

- 4 Show the containers:

```
docker ps
```

- 5 Stop the container:

```
docker stop exx_xml_soap_rpc_server_container_1
```

- 6 Delete the container:

```
docker rm exx_xml_soap_rpc_server_container_1
```

- 7 Remove the image:

```
docker rmi exx_xml_soap_rpc_server_image_1
```

Healthcheck for RPC Server for XML/SOAP

The *docker* directory for RPC Server for XML/SOAP contains a script `healthcheck.sh`. Execution of this script pings the RPC server and returns the result of the ping command:

- 0 success
- 1 ping failure

In the Docker context, this `healthcheck.sh` is put into the Docker container and enabled by setting the `HEALTHCHECK` instruction in the Dockerfile.

You can also use the `healthcheck.sh` script in the context of an orchestration tool (e.g. Kubernetes) to enable healthcheck functionality.

