webMethods EntireX

EntireX RPC Server for IMS Connect

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods EntireX Version 10.5 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 © 1997-2019 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-IMSCONNECT-105-20220422

Table of Contents

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	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.softwareag.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

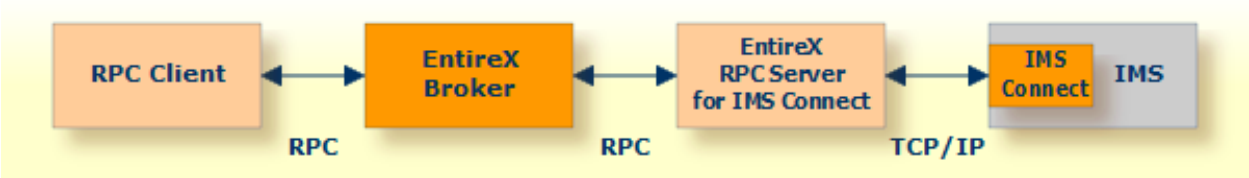
Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to the RPC Server for IMS Connect

The EntireX RPC Server for IMS Connect allows standard RPC clients to communicate with IMS MPP programs. It works together with the IDL Extractor for COBOL and transforms RPC requests from clients into message to IMS, using IMS Connect.

Overview

The RPC Server for IMS Connect acts on one side as an RPC server and on the other side as a client for IMS Connect. The RPC Server for IMS Connect is a Java-based component that can run on a different host to the one where IMS is running. This allows it to operate with a zero footprint of EntireX on the IMS host.

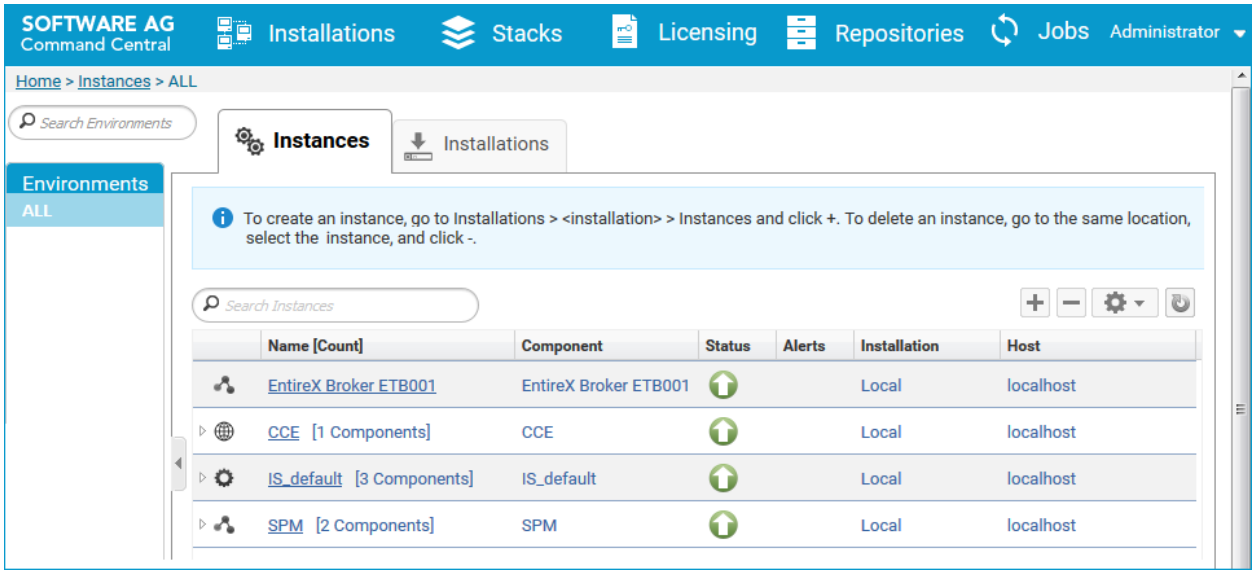


For existing IMS COBOL MPP programs, use the IDL Extractor for COBOL to extract the *Software AG IDL File* in the IDL Editor documentation for the RPC clients.

- For local extraction, all source files have to be stored locally on the same machine where the Designer is running.
- Remote extraction requires an RPC server running under z/OS with Extractor Service (Batch | IMS). See *Step 2: Select a COBOL Extractor Environment or Create a New One* in the IDL Extractor for COBOL documentation.

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).



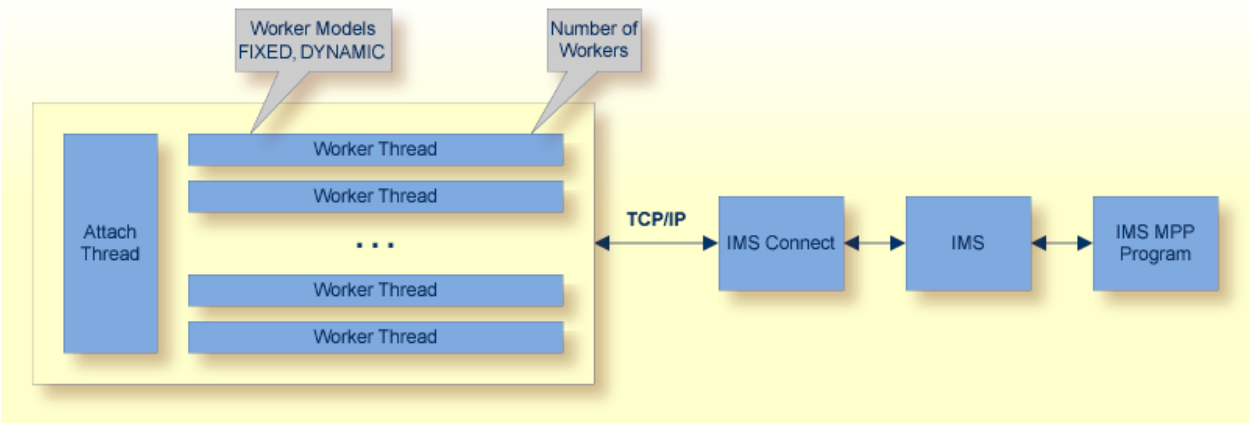
Command Central can assist with the following configuration, management, and monitoring tasks:

- 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 IMS Connect using the Command Central GUI*. For the command-line interface, see *Administering the RPC Server for IMS Connect 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. Every RPC request occupies during its processing a worker thread. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The RPC Server for IMS Connect 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 IMS Connect with Local Scripts*.

3 Administering the RPC Server for IMS Connect using the Command Central GUI

This chapter describes how to administer the EntireX RPC Server for IMS Connect, using the Command Central graphical user interface.

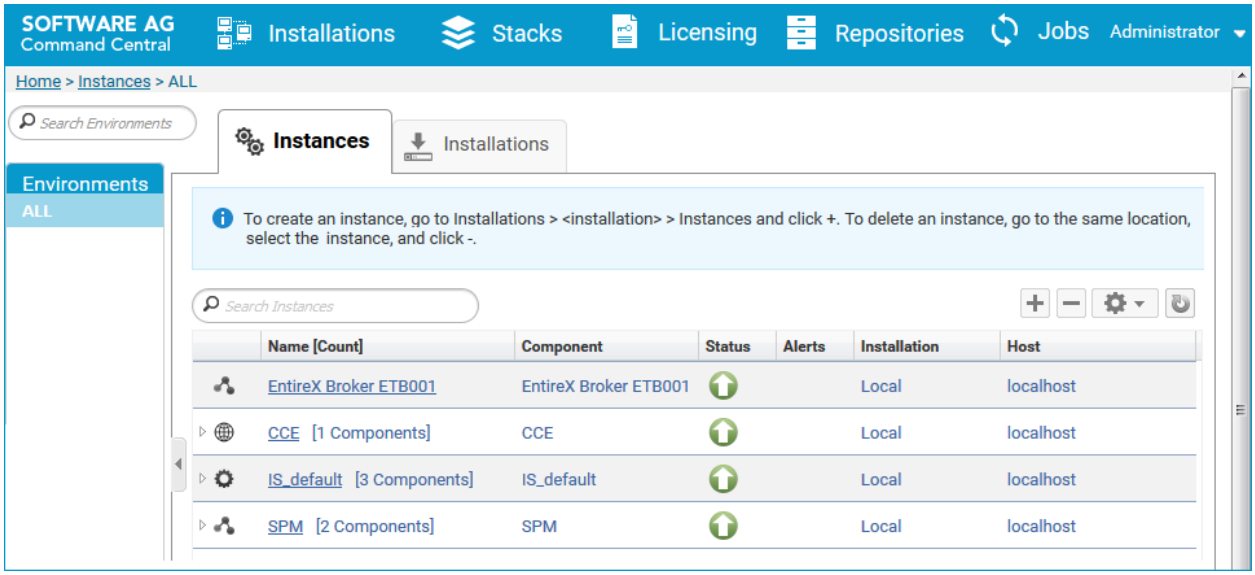
See also *Administering the RPC Server for IMS Connect using the Command Central Command Line*. The core Command Central documentation is provided separately and is also available under **Guides for Tools Shared by Software AG Products** on the Software AG documentation website.

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.

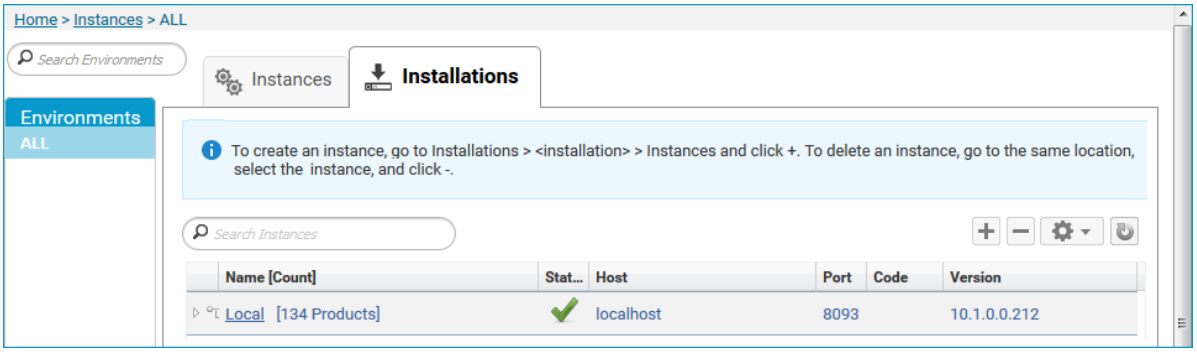
Provide your user credentials in the **Login** page and click **Log In**. This takes you to the page **Home > Instances**:



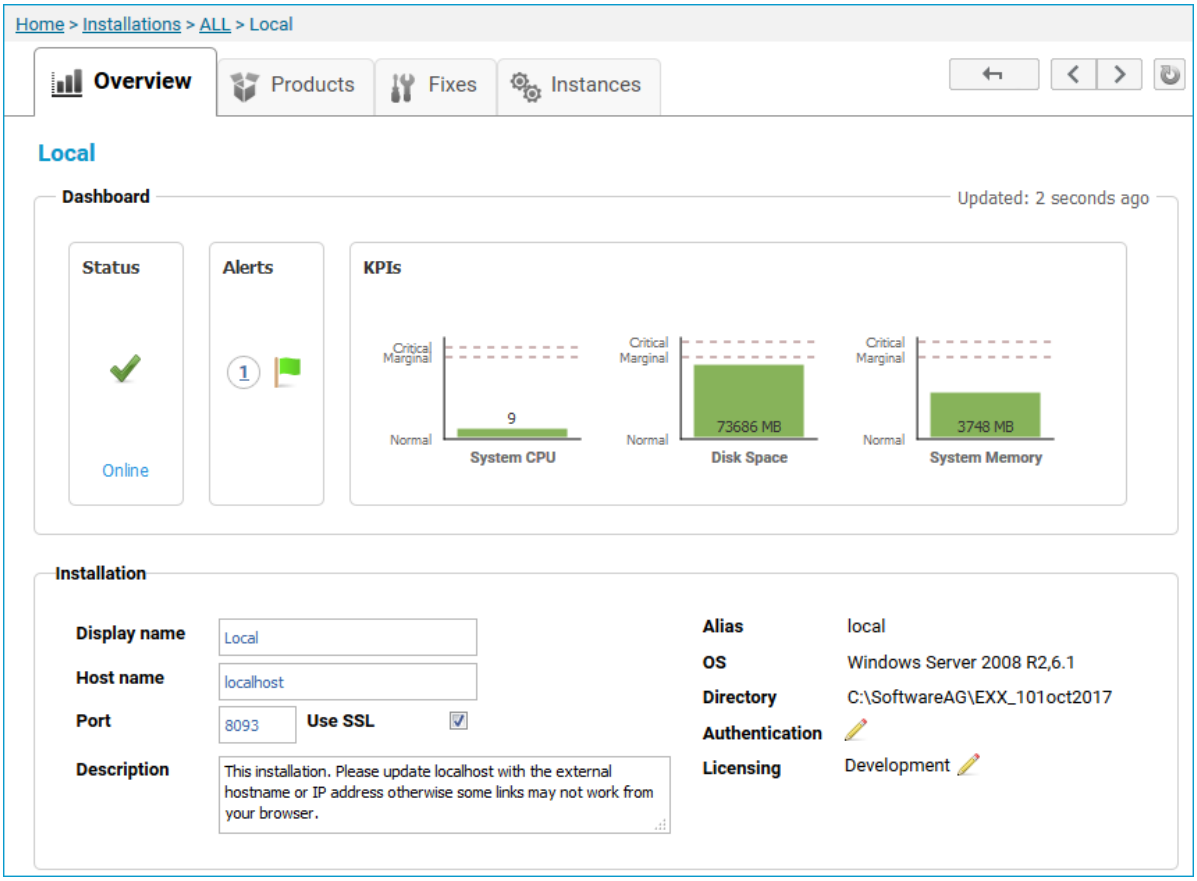
Creating an RPC Server Instance

To create an RPC Server for IMS Connect instance

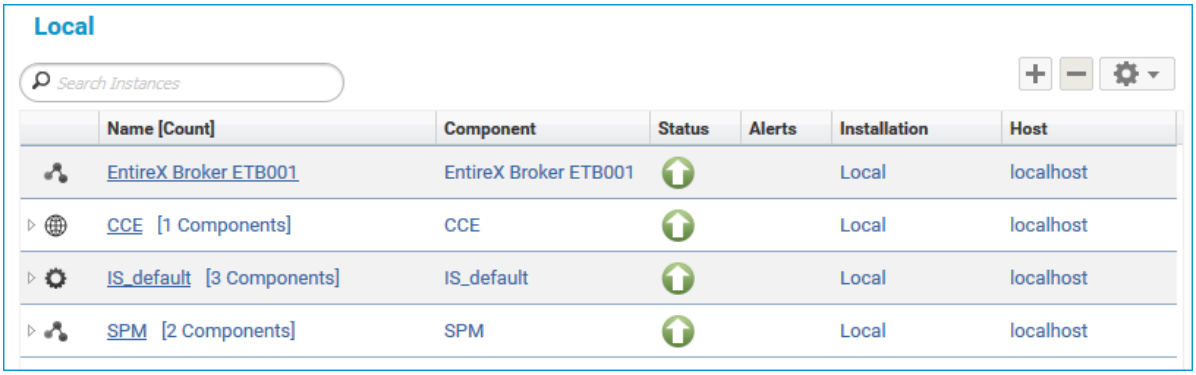
1 In the Command Central home page, click the **Installations** tab.



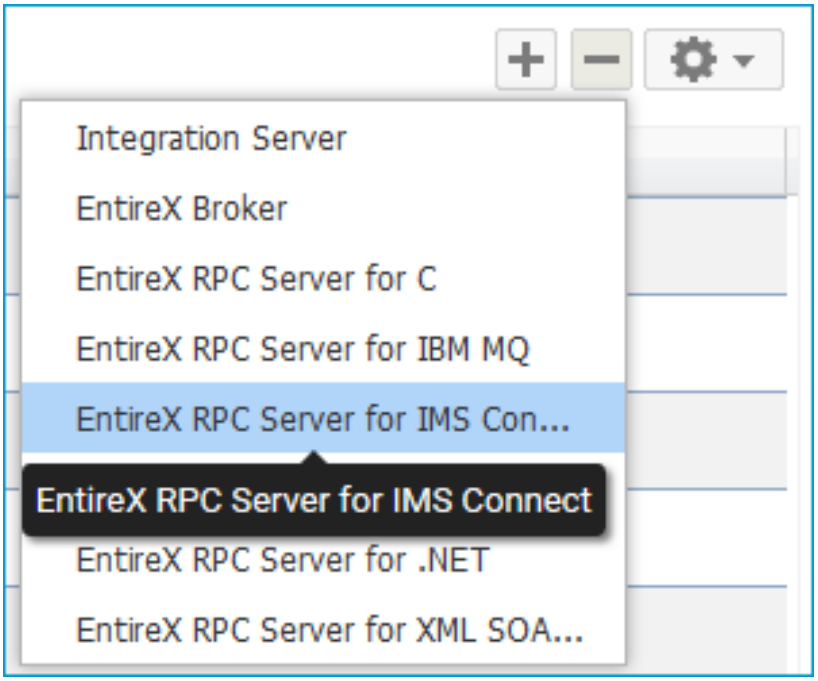
2 Click on the desired installation, for example **Local**, where you want to add an RPC Server for IMS Connect instance.



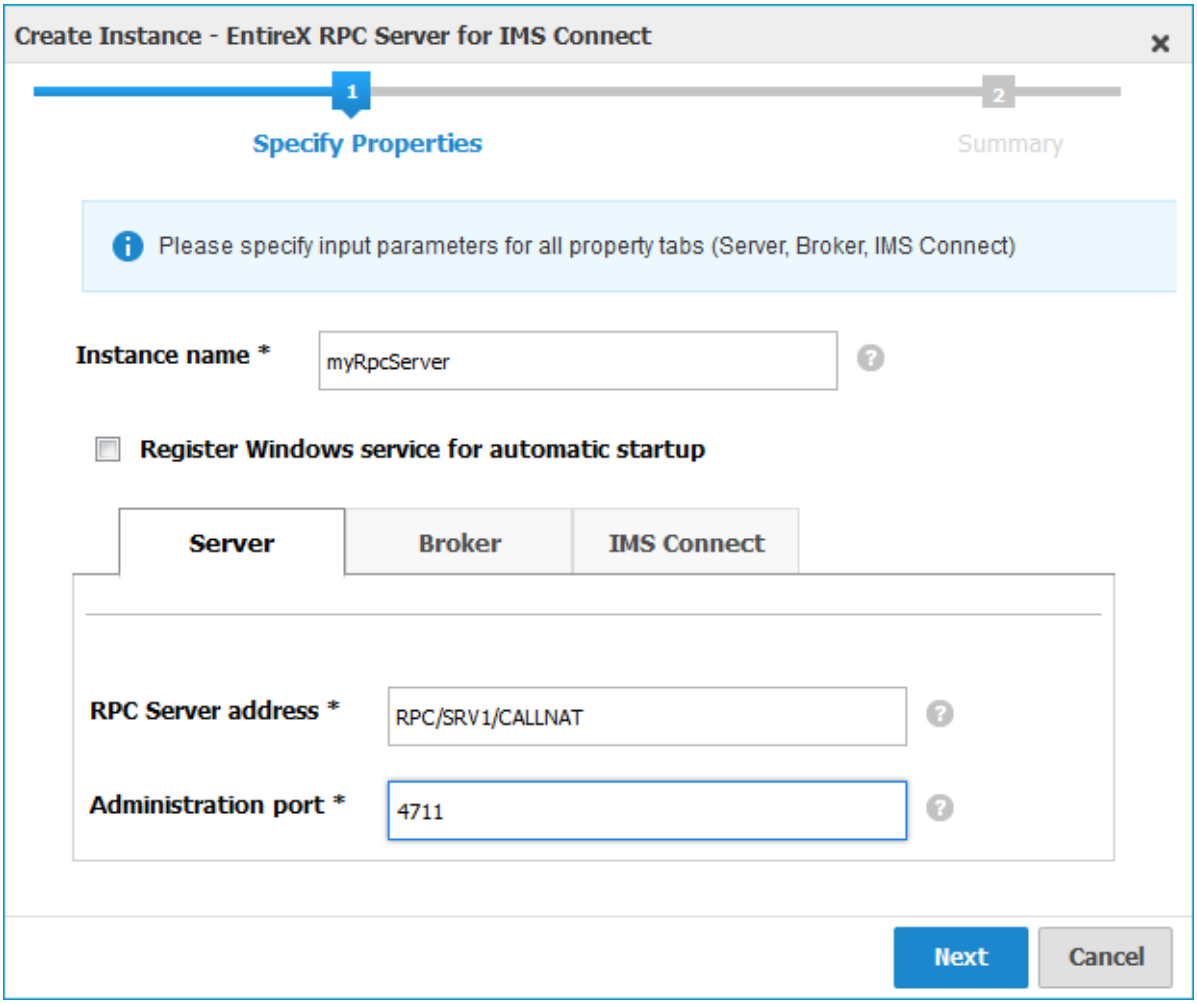
3 Click the **Instances** tab.



4 Click the button in the upper right corner above the list and choose **EntireX RPC Server for IMS Connect**.



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



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.
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.

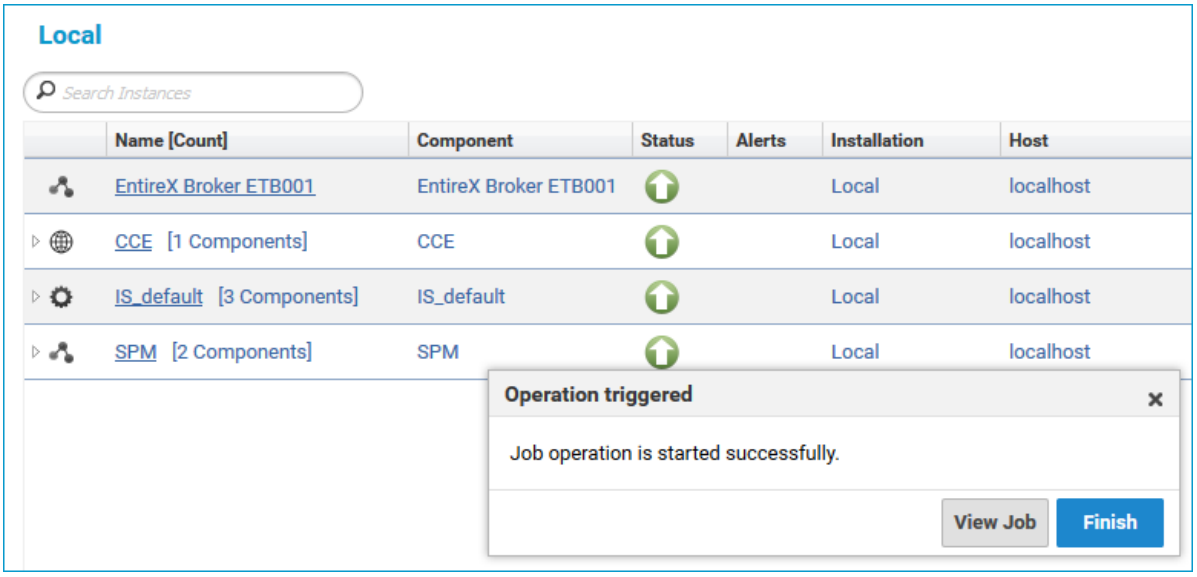
IMS Connect Tab

Here you can modify the IMS Connect settings for the RPC Server for IMS Connect.

Parameter	Description
Connection	
Transport	Required. Use TCP or SSL to communicate with IMS Connect.
IMS host	Required. Host name or IP address where IMS Connect is running.
IMS port	Required. TCP or SSL port number where IMS Connect is listening.
IMS data store ID	Required. Data store ID. Name of the IMS system that will receive transactions.
IMS encoding	Optional. Specify the appropriate EBCDIC encoding used by your IMS Connect. This codepage is also used when communicating with the EntireX Broker.
Credentials	
IMS user	Optional. User ID as defined in your underlying mainframe security system (e.g. RACF).
IMS password	Optional. Password as defined in your underlying mainframe security system (e.g. RACF).

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

7 Press **Finish**.

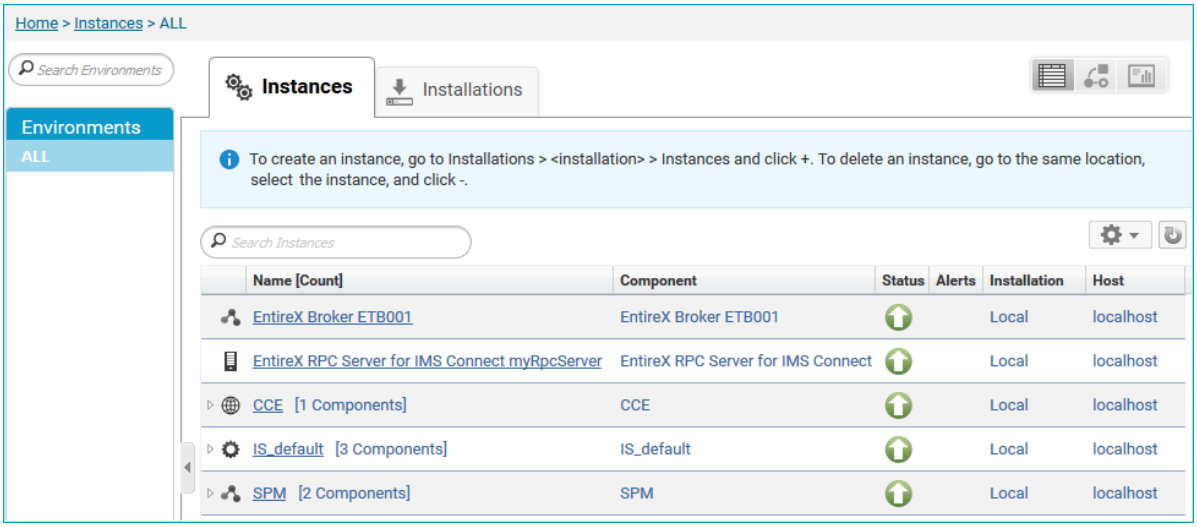


The new instance *myRpcServer* appears in the list.

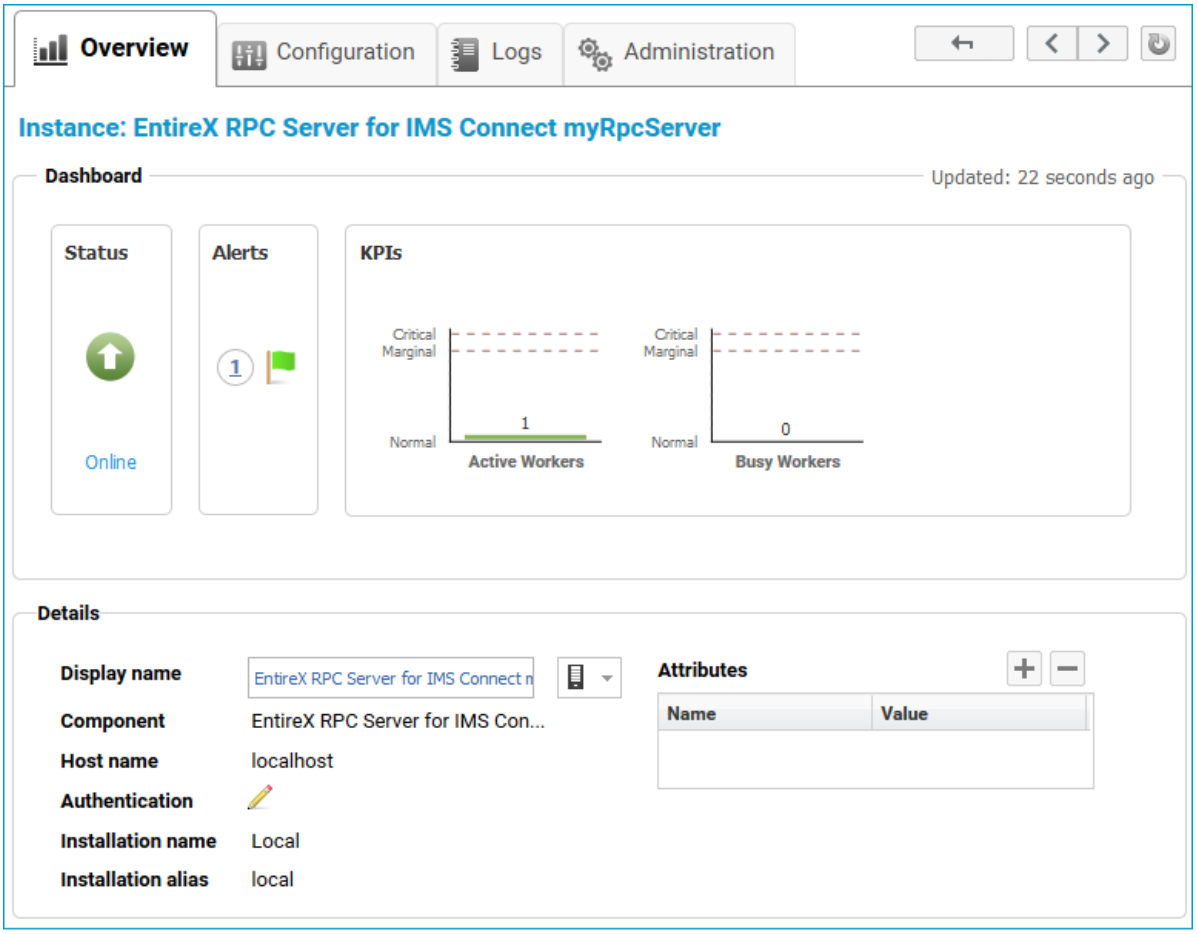
Configuring an RPC Server Instance

To configure an RPC Server for IMS Connect instance

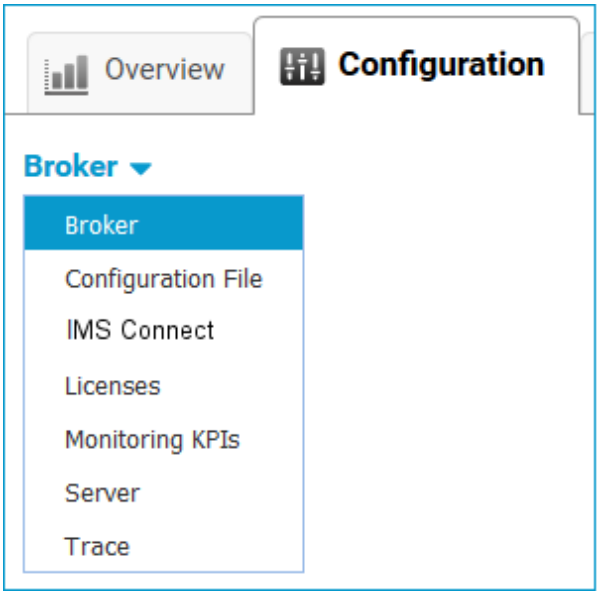
1 In the Command Central home page, click the **Instances** tab.



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



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
- IMS Connect
- Licenses
- Monitoring KPIs
- Server
- Trace Level

Broker

Parameter	Description
Connection	
Transport	Transport over TCP or SSL. Default is `TCP`.
Broker host	Required. EntireX Broker host name or IP address.
Broker port	Required. Port number in range from 1025 to 65535.
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.
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 IMS Connect.

IMS Connect

Here you can modify the IMS Connect settings for the RPC Server for IMS Connect.

Parameter	Description
Connection	
Transport	Required. Use TCP or SSL to communicate with IMS Connect.
IMS host	Required. Host name or IP address where IMS Connect is running.
IMS port	Required. TCP or SSL port number where IMS Connect is listening.
IMS data store ID	Required. Data store ID. Name of the IMS system that will receive transactions.
IMS encoding	Optional. Specify the appropriate EBCDIC encoding used by your IMS Connect. This codepage is also used when communicating with the EntireX Broker.
IMS socket timeout	Optional. Socket timeout for connection to IMS Connect (in milliseconds).
IMS SSL trust store	Optional. Specifies the location of SSL trust store.
IMS SSL verify server	Optional. The RPC Server as SSL client checks the identity of IMS Connect as SSL server.
Credentials	
IMS user	Optional. User ID as defined in your underlying mainframe security system (e.g. RACF).
IMS password	Optional. Password as defined in your underlying mainframe security system (e.g. RACF).
Exit	
IMS Connect exit	Required. Use Old or New.
IMS Connect exit name	Optional. If left blank, the following defaults will be used: *SAMPLE* for old exit, *SAMPLE1* for new exit.

Licences

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 IMS Connect: Active Workers and Busy Workers.

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for IMS Connect. 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 IMS Connect stops.
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 IMS Connect.

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.

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 IMS Connect instance for which you want to see the runtime status (same as Step 1 under *Configuring a Broker Instance*).

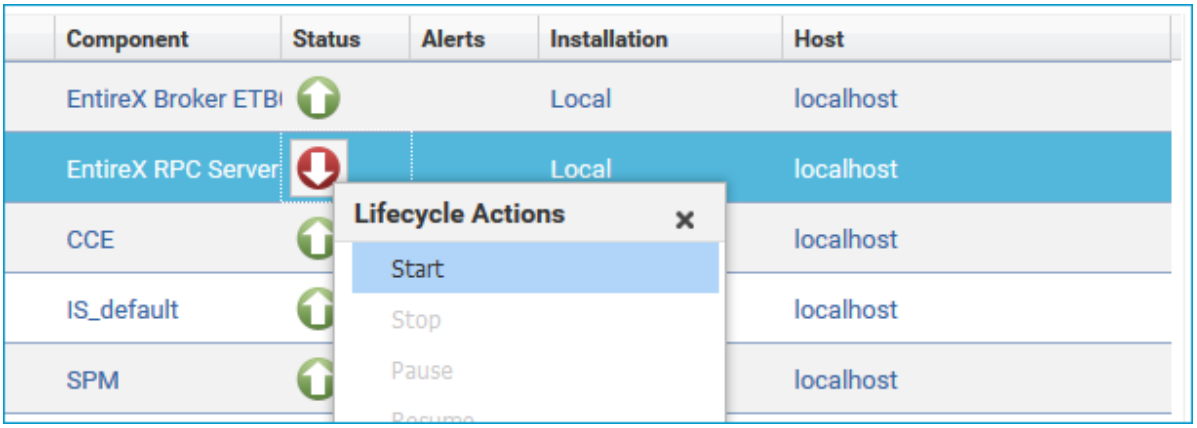
The visual key performance indicators (KPIs) and alerts enable you to monitor the RPC Server for IMS Connect's health.

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 IMS Connect instance from the Instances tab

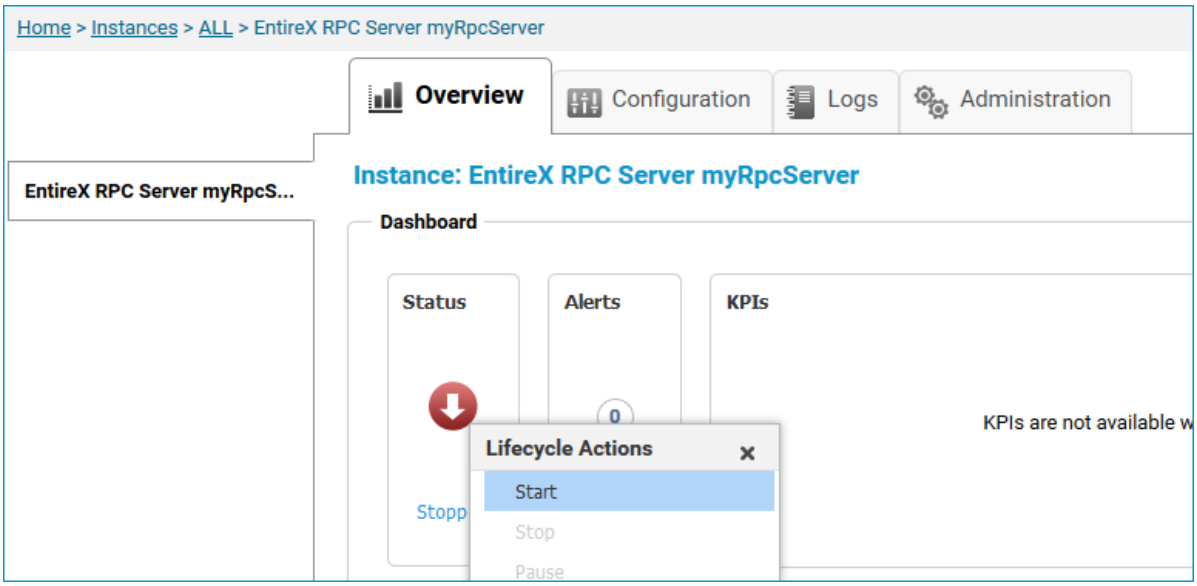
1 In the Command Central home page, click the **Instances** tab.



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

To start an RPC Server for IMS Connect instance from its Overview tab

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

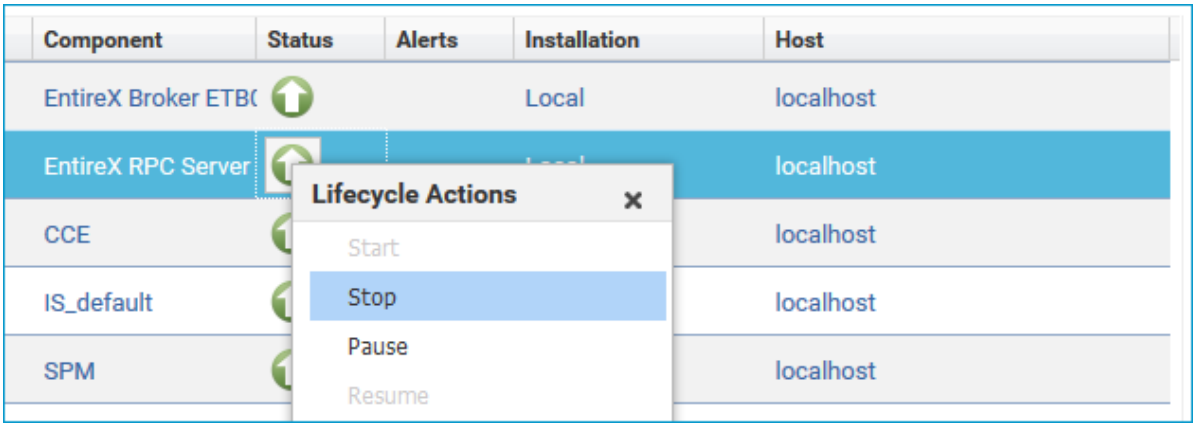


2 Select the status, and from the context menu choose **Start**.

Stopping an RPC Server Instance

To stop an RPC Server for IMS Connect instance from the Instances tab

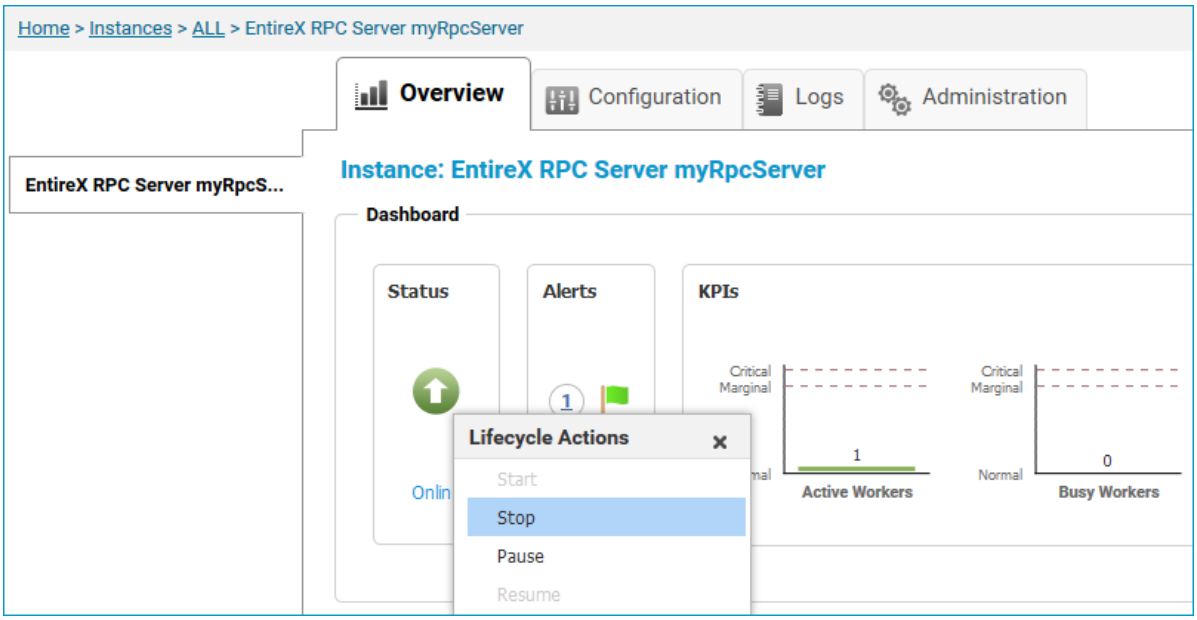
1 In the Command Central home page, click the **Instances** tab.



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

To stop an RPC Server for IMS Connect instance from its Overview tab

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



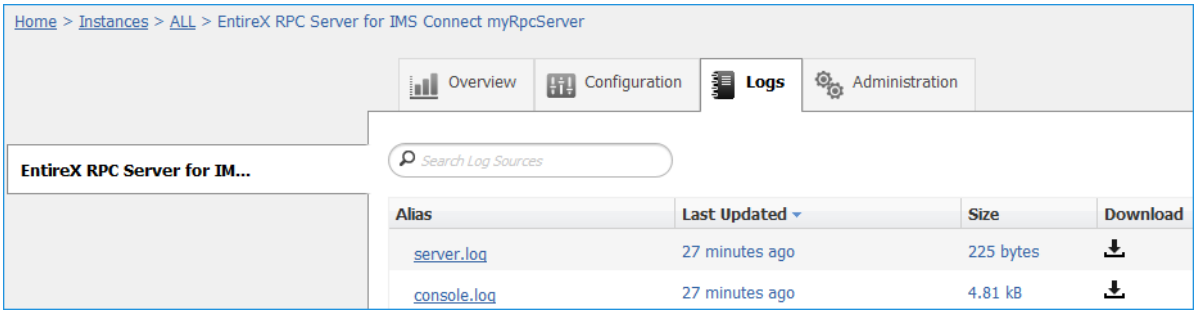
2 Select the status, and from the context menu choose **Stop**.

Inspecting the Log Files

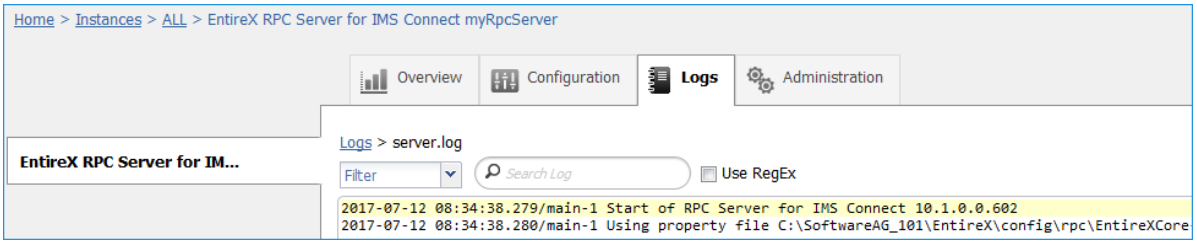
To inspect the log files of an RPC Server for IMS Connect instance

1 In the Command Central home page, click the **Instances** tab, then click the link associated with the RPC Server for IMS Connect 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*:

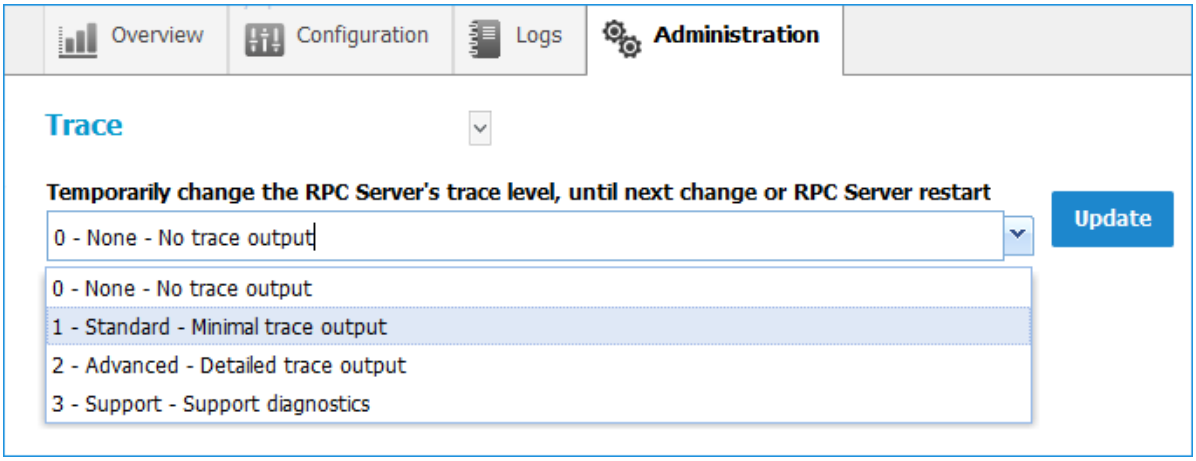


Changing the Trace Level Temporarily

To temporarily change the trace level of an RPC Server for IMS Connect instance

1 In the Command Central home page, click the **Instances** tab then click the link associated with the RPC Server for IMS Connect 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**.

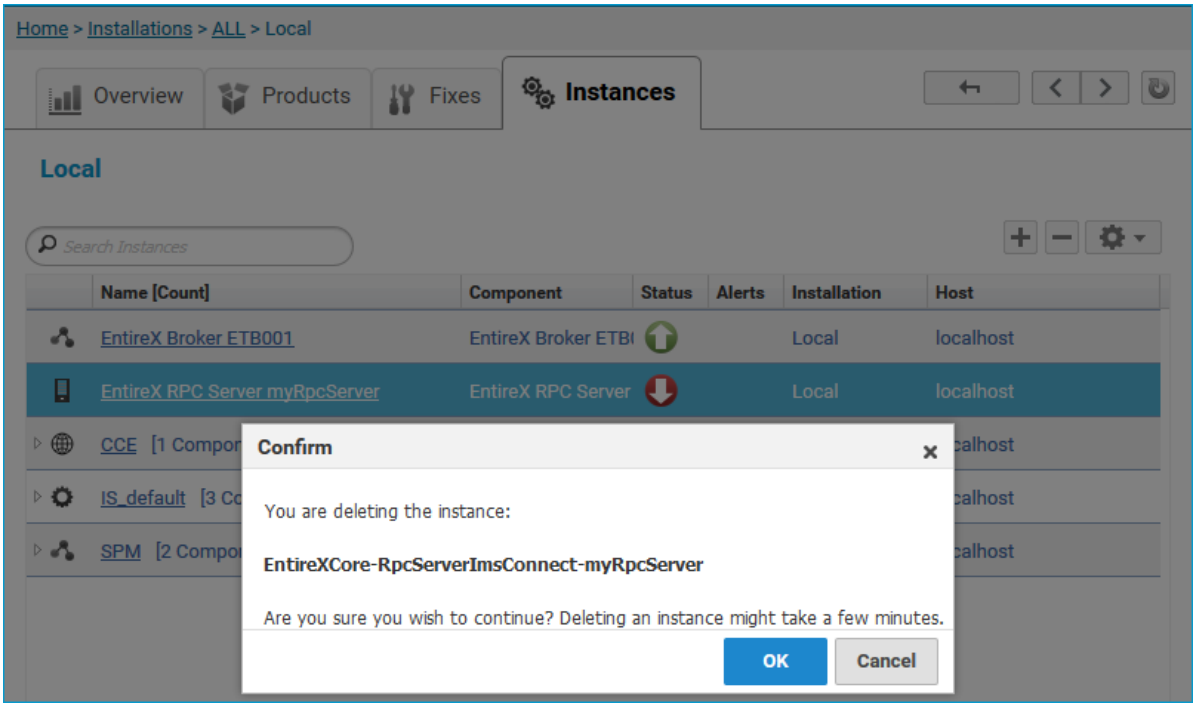


Note: If you want to set the trace level permanently, see *Trace Level* under *Configuring an RPC Server Instance*.

Deleting an RPC Server Instance

To delete an RPC Server for IMS Connect instance

1 In the list of EntireX RPC Server for IMS Connect 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 IMS Connect instance.

3 In the next window, click **Finish**. The selected instance is removed from the list.

4 Administering the RPC Server for IMS Connect using the Command Central Command Line

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

Administering the RPC Server for IMS Connect using the Command Central GUI is described under *Administering the RPC Server for IMS Connect 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`	*`node_alias`*	*`name`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`type`	`RpcServerImsConnect`	Required. EntireXCore instance type of RPC server. Must be "`RpcServerImsConnect`".
	`product`	`EntireXCore`	Required. Must be set to "`EntireXCore`".
	`instance.name`	*`name`*	Required. Name of the runtime component, for example "MyRpcServer".
	`install.service`	`true` | `false`	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.
	`server.address`	*`class/server/service`*	Required. The case-sensitive RPC server address has the format: `CLASS/SERVER/SERVICE`.
	`server.adminport`	`1025-65535`	Required. The administration port in range from 1025 to 65535.
	`broker.transport`	`ssl` | `tcp`	Transport over TCP or SSL. Default is `TCP`.
	`broker.host`	*`name`*	Required. EntireX Broker host name or IP address.
	`broker.port`	`1025-65535`	Required. Port number in range from 1025 to 65535.
	`broker.user`	*`user`*	Optional. The user ID for secured access to the broker.
	`broker.password`	*`password`*	Optional. The password for secured access to the broker.

Command	Parameter	Value	Description
	`ims.transport`	`ssl | tcp`	Required. Use TCP or SSL to communicate with IMS Connect.
	`ims.host`	*name*	Required. Host name or IP address where IMS Connect is running.
	`ims.port`	*n*	Required. TCP or SSL port number where IMS Connect is listening.
	`ims.datastoreid`	*name*	Required. Data store ID. Name of the IMS system that will receive transactions.
	`ims.encoding`	*codepage*	Optional. Specify the appropriate EBCDIC encoding used by your IMS Connect. This codepage is also used when communicating with the EntireX Broker.
	`ims.user`	*userid*	Optional. User ID as defined in your underlying mainframe security system (e.g. RACF).
	`ims.password`	*password*	Optional. Password as defined in your underlying mainframe security system (e.g. RACF).

Example

- To create a new instance for an installed EntireX of the type "RpcServerImsConnect", with name "MyRpcServer", with server address "RPC/SRV1/CALLNAT", using administration port 5757, with broker host name "localhost", listening on broker port 1971, using IMS transport TCP on host "IMSHOST", connected with port 3838, using IMS system "IMSSYS" and encoding "cp037" in the installation with alias name "local":

```
sagcc create instances local EntireXCore type=RpcServerImsConnect
instance.name=MyRpcServer server.address=RPC/SRV1/CALLNAT server.adminport=5757
broker.host=localhost broker.port=1971 ims.transport=TCP ims.host=IMSHOST
ims.port=3838 ims.datastoreid=IMSSYS ims.encoding=cp037
```

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

Configuring an RPC Server Instance

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

- Broker
- Configuration File
- IMS Connect

- Monitoring KPIs
- Server
- Trace Level

Broker

Here you can administer the parameters used for communication between the RPC Server for IMS Connect 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.
`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.

Displaying the Broker Settings of the RPC Server

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

Example 1

- To display the Broker parameters of the RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local EntireXCore-RpcServerImsConnect-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-RpcServerImsConnect-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"
}
```

Updating the Broker Settings of the RPC Server

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

Example

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

```
sagcc update configuration data local EntireXCore-RpcServerImsConnect-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 IMS Connect. 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXCore-RpcServerImsConnect-`".
	`instanceid`	Required. Must be "`CONFIGURATION`".
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

- To display the configuration file of the RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local":

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

Updating the Content of the RPC Server Configuration File

Command	Parameter	Description
`sagcc update configuration data`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXCore-RpcServerImsConnect-`".
	`instanceid`	Required. Must be "`CONFIGURATION`".
	`-i` *`file`*	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-RpcServerImsConnect-MyRpcServer
CONFIGURATION -i configuration.json
```

IMS Connect

Here you can modify the IMS Connect settings for the RPC Server for IMS Connect.

- Parameters
- Displaying the IMS Connect Parameters
- Updating the IMS Connect Parameters

Parameters

Parameter	Description
`ImsTransport`	Use TCP or SSL to communicate with IMS Connect.
`ImsHost`	Host name or IP address where IMS Connect is running.
`ImsPort`	TCP or SSL port number where IMS Connect is listening.
`ImsDataStoreId`	Data store ID. Name of the IMS system that will receive transactions.
`ImsEncoding`	Specify the appropriate EBCDIC encoding used by your IMS Connect. This codepage is also used when communicating with the EntireX Broker.
`ImsUser`	User ID as defined in your underlying mainframe security system (e.g. RACF).
`ImsPassword`	Password as defined in your underlying mainframe security system (e.g. RACF).
`ImsSslTrustStore`	Specifies the location of SSL trust store.
`ImsSslVerifyServer`	The RPC Server as SSL client checks the identity of IMS Connect as SSL server.
`ImsTimeout`	Socket timeout for connection to IMS Connect (in milliseconds).
`ImsUseOldExit`	Control the default exit name. True: `*SAMPLE*`, False: `*SAMPLE1*`.
`ImsExitName`	If left blank, the following defaults will be used: `*SAMPLE*` for old exit, `*SAMPLE1*` for new exit.

Displaying the IMS Connect Parameters

Command	Parameter	Description
`sagcc get configuration data`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"IMS-CONNECT"`.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

- To display the IMS Connect parameters of RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local EntireXCore-RpcServerImsConnect-MyRpcServer
IMS-CONNECT
```

Example 2

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

```
sagcc get configuration data local EntireXCore-RpcServerImsConnect-MyRpcServer
IMS-CONNECT -o ims_connect.json
```

Resulting output file in JSON format:

```
{
    "ImsTransport": "TCP",
    "ImsHost": "IMSHOST",
    "ImsPort": "3838",
    "ImsDataStoreId": "IMSSYS",
    "ImsEncoding": "cp037",
    "ImsUser": "",
    "ImsPassword": "",
    "ImsSslTrustStore": "",
    "ImsSslVerifyServer": "true",
    "ImsTimeout": "10000",
    "ImsUseOldExit": "true",
    "ImsExitName": ""
}
```

Updating the IMS Connect Parameters

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

Example

- To load the IMS Connect parameters of the RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local" from file *http.json* in the current working directory:

```
sagcc update configuration data local EntireXCore-RpcServerImsConnect-MyRpcServer
IMS-CONNECT -i ims_connect.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 IMS Connect: Active Workers and Busy Workers.

Parameters

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for IMS Connect. 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"EXX-MONITORING-KPIS"`.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

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

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

Resulting output file in text format:

```
entirex.entirex.spm.version=10.5.0.0.473
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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXCore-RpcServerImsConnect-`".
	`instanceid`	Required. Must be "`EXX-MONITORING-KPIS`".
	`-i` *`file`*	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-RpcServerImsConnect-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 IMS Connect.

Parameters

Parameter	Value	Description
`ServerAddress`	*`class/server/service`*	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`	*`n`*	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 IMS Connect stops.
`WorkerScalability`	*`true`* | *`false`*	You can either have a fixed or dynamic number of workers. Default is `dynamic` (`true`). 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"SERVER"`.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

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

```
sagcc get configuration data local EntireXCore-RpcServerImsConnect-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-RpcServerImsConnect-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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"SERVER"`.
	`-i` *`file`*	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-RpcServerImsConnect-MyRpcServer
SERVER -i server.json
```

See **Example 2** above for sample input file.

Trace Level

Here you can set the trace level of the RPC Server for IMS Connect.

- 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"TRACE"`.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

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

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

Resulting output file in JSON format:

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

Updating the Trace Level

Command	Parameter	Description
`sagcc update configuration data`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`instanceid`	Required. Must be `"TRACE"`.
	`-i` *`file`*	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-RpcServerImsConnect-MyRpcServer
TRACE -i trace.json
```

See **Example 2** above for sample input 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

Example

- To display state information about the RPC Server for IMS Connect:

```
sagcc get monitoring state local EntireXCore-RpcServerImsConnect-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 IMS Connect, using the Command Central `exec lifecycle` commands.

Command	Parameter	Description
`sagcc exec lifecycle start`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

Example

- To start the RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle start local EntireXCore-RpcServerImsConnect-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 IMS Connect, using the Command Central `exec lifecycle` commands.

Command	Parameter	Description
`sagcc exec lifecycle stop`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

Example

- To stop the RPC Server for IMS Connect "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle stop local EntireXCore-RpcServerImsConnect-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 IMS Connect. 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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

Example

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

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

Getting Content from or Downloading RPC Server Log Files

Command	Parameter	Description
`sagcc get diagnostics logs`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`full | tail | head`	Optional. Shows full log file content, or only tail or head.
	`export -o` *`file`*	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-RpcServerImsConnect-MyRpcServer
server.log tail
```

Example 2

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

```
sagcc get diagnostics logs local EntireXCore-RpcServerImsConnect-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 Level*** 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 IMS Connect "MyRpcServer" in the installation with alias name "local" in JSON format on stdout:

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

Example 2

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

```
sagcc exec administration component local
EntireXCore-RpcServerImsConnect-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.
	`node_alias`	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.
	`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-RpcServerImsConnect-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`	*`node_alias`*	Required. Specifies the alias name of the installation in which the runtime component is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXCore-RpcServerImsConnect-"`.

Example

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

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

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

5 Administering the RPC Server for IMS Connect with Local Scripts

The EntireX RPC Server for IMS Connect allows standard RPC clients to communicate with IMS MPP programs. It works together with the IDL Extractor for COBOL and transforms RPC requests from clients into message to IMS, using IMS Connect.

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

See also *Administering the RPC Server for IMS Connect* with the Command Central **GUI** | **Command Line**.

Customizing the RPC Server

The following elements are used to set up the RPC Server for IMS Connect:

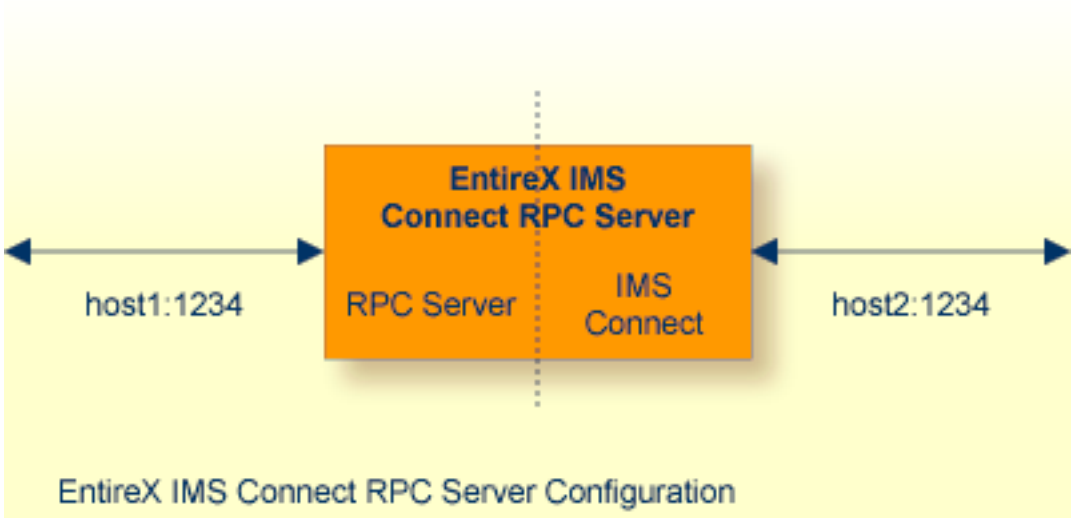
- Configuration File
- Start Script

Configuration File

The default name of the configuration file is *entirex.imsconnect.properties*. The RPC Server for IMS Connect searches for this file in the current working directory.

You can set the name of the configuration file with `-Dentirex.server.properties=<your file name>` with "/" as file separator.

The configuration file contains the configuration for both parts of the RPC Server for IMS Connect.



Configuring more than one RPC Server

If you configure more than one RPC Server for IMS Connect that connects to the same broker, the following items must be distinct:

- the trace output file (property `entirex.server.logfile`)
- the log for the Windows Service (property `entirex.server.serverlog`)

Start Script

The start script for the RPC Server for IMS Connect is called *imsconnectserver.bsh* (UNIX) or *imsconnectserver.bat* (Windows) and is provided in the *bin* folder of the installation directory. You may customize this file. The RPC Server for IMS Connect itself is contained in the file *entirex.jar*.

Configuring the RPC Server Side

The RPC Server for IMS Connect 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. See *URL-style Broker ID* in the EntireX Broker ACI Programming documentation.
`entirex.server.serveraddress`	`-server`	`RPC/SRV1/CALLNAT`	Server address.
`entirex.server.userid`	`-user`	`IMSRPCServer`	The user ID for access to the broker.
`entirex.server.fixedservers`		`no`	`NO` The number of worker threads balances between what is specified in `entirex.server.minservers` and what is specified in `entirex.server.maxservers`. This is done by a so-called attach thread. At startup, the number of worker threads is the number specified in `entirex.server.minservers`. A new worker thread starts if the broker has more requests than there are worker threads waiting. If more than the number specified in `entirex.server.minservers` are waiting for requests, a worker thread stops if its receive call times out. The timeout period is configured with `entirex.server.waitserver`. See worker model `DYNAMIC`. `YES` The number of worker threads specified in `entirex.server.minservers` is started and the server can process this number of parallel requests. See worker model `FIXED`.
`entirex.server.minservers`		`1`	Minimum number of server threads.

Property Name	Command-line Option	Default	Explanation
entirex.server.maxservers		32	Maximum number of server threads.
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 IMS Connect running while the Broker is down for a short time.
entirex.server.password	-password		The password for secured access to the broker. The password is encrypted and written to the property entirex.server.password.e. To change the password, set the new password in the properties file. To disable password encryption, set entirex.server.passwordencrypt=no. Default: yes.
entirex.server.properties	-propertyfile	entirex.server. properties	The file name of the property file.
entirex.server.security	-security	no	no|yes|auto|name of BrokerSecurity object
entirex.server. compresslevel	-compresslevel	0	Permitted values (you can enter the text or the numeric value) BEST_COMPRESSION 9 BEST_SPEED 1 DEFAULT_COMPRESSION -1 (*mapped to* 6) DEFLATED 8 NO_COMPRESSION 0 N 0 Y 8
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. See *Setting the Transport Timeout* under *Writing Advanced Applications - EntireX Java ACI*.
	-help		Display usage of the command-line parameters.
entirex.server.logfile	-logfile		Name of the log file, the default is standard output.
entirex.trace	-trace	0	Trace level (1,2,3).

Configuring the IMS Connect Side

These properties are used to configure the connection to IMS Connect.

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

Name	Default Value	Explanation
`ims.host`		Host name of IMS Connect. Mandatory.
`ims.port`		Port number of IMS Connect. Mandatory.
`ims.datastoreid`		Data store ID. Name of the IMS system that will receive transactions. Mandatory.
`entirex.bridge.targetencoding`	`cp037`	Specify the appropriate EBCIDIC encoding used by your IMS Connect. This codepage is also used when communicating with the EntireX Broker. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to `"SAGTRPC"`. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*. Default "cp037" is EBCDIC codepage with full Latin-1 character set.
`ims.useoldexit`	`yes`	yes Use old IMS Connect user message exit. Name is *SAMPLE*. no Use new IMS Connect user message exit. Name is *SAMPLE1*.
`ims.exitname`	`*SAMPLE*` (old exit) `*SAMPL1*` (new exit)	Name of IMS Connect user message exit.
`ims.sockettimeout`	`10000`	Socket timeout for connection to IMS Connect (in milliseconds).
`ims.checkdfs`	true	true, yes Check for DFS message. Return an error and do not return the message if it contains a DFS error message. false, no Do not check for DFS message.

Name	Default Value	Explanation
`ims.clientid`		ID of the client that is used by IMS Connect. Maximum 8 bytes (optional).
`ims.lterm`		IMS `LTERM` override. Maximum 8 bytes (optional).
`ims.userid`		RACF user ID. Maximum 8 bytes (optional).
`ims.groupid`		RACF group ID. Maximum 8 bytes (optional).
`ims.password`		RACF password/PassTicket. Maximum 8 bytes (optional).
`ims.applname`		RACF application name. Maximum 8 bytes (optional).
`ims.sslparams`		SSL parameters (optional). Same syntax as Broker ID.
`ims.mapping.folder`		The folder where the RPC server expects server-side mapping files (Designer files with extension .svm). See ***Deploying Server-side Mapping Files to the RPC Server*** and ***Undeploying Server-side Mapping Files from the RPC Server***. There are also client-side mapping files that do not require configuration here. See *Server Mapping Files for COBOL*. If *no* server requires server-side mapping, you can omit this property. If *one* server requires server-side mapping, this property must be specified.
`ims.useprogramname`	`false`	Automatically use the IDL program name as transaction name. If set to "true" or "yes", 10 bytes are used for the transaction name. If set to a number, this number of bytes is used for the transaction name.

Using SSL/TLS with the RPC Server

To use SSL with the RPC Server for IMS Connect, you need to configure two sides:

- **IMS Connect Side**
  See parameter `ims.sslparams` under ***Configuring the IMS Connect Side***.

- **RPC Server Side**
  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 and Certificates with EntireX* in the Platform-independent Administration documentation.

To use SSL

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 IMS Connect 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 side 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 UNIX and Windows Administration documentation

Starting the RPC Server

To start the RPC Server for IMS Connect

- Use the ***Start Script***.

  Or:

  Under Windows you can use the RPC Server for IMS Connect as a Windows Service. See ***Running an EntireX RPC Server as a Windows Service***.

Stopping the RPC Server

To stop the RPC Server for IMS Connect

- 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 IMS Connect

- 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.

Running an EntireX RPC Server as a Windows Service

For general information see *Running an EntireX RPC Server as a Windows Service*.

To run the RPC Server for IMS Connect as a Windows Service

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

Note: The script must pass external parameters to the RPC server and use the reduced signaling of the JVM (option `-Xrs`):

```
java -Xrs com.softwareag.entirex.ims.IMSRPCServer %*
```

If `-Xrs` is not used, the JVM stops and an entry 10164002 is written to the event log when the user logs off from Windows.

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 *imsconnectserver.bat*, the command will be

```
RPCService -install -ext MyServer ↩
-script install_path\EntireX\bin\imsconnectserver.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".

Application Identification

The application identification is sent from the RPC Server for IMS Connect to the Broker. It is visible with Broker Command and Info Services.

The identification consists of four parts: name, node, type, and version. These four parts are sent with each Broker call and are visible in the trace information.

For the RPC Server for IMS Connect, these values are:

Identification Part	Value
Application name	`ANAME=IMS Connect RPC Server`
Node name	`ANODE=<host name>`
Application type	`ATYPE=Java`
Version	`AVERS=10.5.0.0`

6 Extracting from Message Format Service

To extract interface definitions from Message Format Service (MFS) with MID and MOD definitions, use a command-line extractor. Run the extractor with the following command:

```
java -classpath <suite installation ↩
folder>\IntegrationServer\packages\WmEntireX\code\jars\entirex.jar ↩
com.softwareag.entirex.ims.extractor.MFSExtractor <inputfile>
```

The input file must be an MFS source with MID and MOD definitions. The output is an IDL file with the same name as the input file and suffix ".idl". Use the IDL file to generate connections and adapter services with the Integration Server Wrapper.

7 Server-side Mapping Files

Server mapping enables the RPC server to correctly support special COBOL syntax such as `REDEFINE`s, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the IDL Extractor for COBOL automatically extracts a server mapping file in addition to the IDL file (interface definition language). Also, the COBOL Wrapper may generate a server mapping file for RPC server generation. The server mapping is used at runtime to marshal and unmarshal the RPC data stream. There are client-side mapping files (Designer files with extension .cvm) and server-side mapping files (Designer files with extension .svm). If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL* in the Designer documentation.

See also *Source Control of Server Mapping Files* | *Comparing Server Mapping Files* | *When is a Server Mapping File Required?* | *Migrating Server Mapping Files* in the Designer documentation.

Server-side Mapping Files in the RPC Server

For an RPC Server for IMS Connect, server mapping information is contained in a server-side mapping file (Designer file with extension .svm) See *Server Mapping Files for COBOL*. Server mapping files are provided as operating system files in an RPC server related server-side mapping container (directory or folder). The files have the same format as in the Designer. See ***Configuring the IMS Connect Side***.

If *no* server requires a server mapping file, you can omit the property `ims.mapping.folder`.

If *one* server requires a server mapping file, provide the property `ims.mapping.folder`.

See also ***Deploying Server-side Mapping Files to the RPC Server***.

Deploying Server-side Mapping Files to the RPC Server

Deploy a server-side mapping file (Designer file with extension .svm) to the RPC server manually. See *Server Mapping Files for COBOL* in the Designer documentation.

To deploy a server-side mapping file

1. Make sure the server-side mapping container (directory or folder) is configured. See ***Server-side Mapping Files in the RPC Server***.
2. Copy the server-side mapping file to the server-side mapping container.

Undeploying Server-side Mapping Files from the RPC Server

Undeploy a server mapping file (Designer file with extension .svm) from the RPC server manually. See *Server Mapping Files for COBOL*.

To undeploy a server-side mapping file manually

- Delete the server-side mapping file from the server-side mapping container (directory or folder). See *Server Mapping Files for COBOL*.

Change Management of Server-side Mapping Files

Under UNIX and Windows, change management for a directory or folder (server-side mapping container, see ***Server-side Mapping Files in the RPC Server***) is similar to change management within ordinary operating system directories (folders). All updates to the directory or folder done after a backup must be kept.

All Designer server-side mapping files (.svm) added since the last backup should be available. See *Server Mapping Files for COBOL* in the Designer documentation.

List Deployed Server-side Mapping Files

Use the Windows Explorer or the UNIX `ls` command to list the contents of the server-side mapping container (directory or folder). See ***Server-side Mapping Files in the RPC Server***.

Check if a Server-side Mapping File Revision has been Deployed

Server-side mapping files in the server-side mapping container correspond to Designer files with extension .svm (same format). See *Server Mapping Files for COBOL* in the Designer documentation. Each line relates to an IDL program and contains a creation timestamp at offset 276 (decimal) in the format *YYYYMMDDHHIISST*. Precision is 1/10 of a second. The creation timestamp can be checked.

The timestamp can be found on the same offset in the server-side mapping files stored in the server-side mapping container (directory or folder). See ***Server-side Mapping Files in the RPC Server***.

Is There a Way to Smoothly Introduce Server-side Mapping Files?

All EntireX RPC servers can be executed without server-side mapping files. See ***Server-side Mapping Files in the RPC Server***. There is no need to install the server-side mapping container if the following conditions are met:

- You do not use features that require server mapping; see *When is a Server Mapping File Required?*
- Server-side type of COBOL mapping is switched on in the Designer. If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL*.

You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires server-side mapping. All EntireX RPC servers are backward compatible.

8 Scenarios

COBOL Scenarios

Scenario I: Calling an Existing COBOL Server

To call an existing COBOL server

1 Use the IDL Extractor for COBOL to extract the Software AG IDL and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* in the Designer documentation.

2 Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:

- use the IDL Tester; see *EntireX IDL Tester* in the Designer documentation
- generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Server Examples for z/OS IMS MPP* in the COBOL Wrapper documentation for COBOL RPC Server examples.