webMethods EntireX

EntireX RPC Server for CICS Socket Listener

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-CICSSOCKET-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

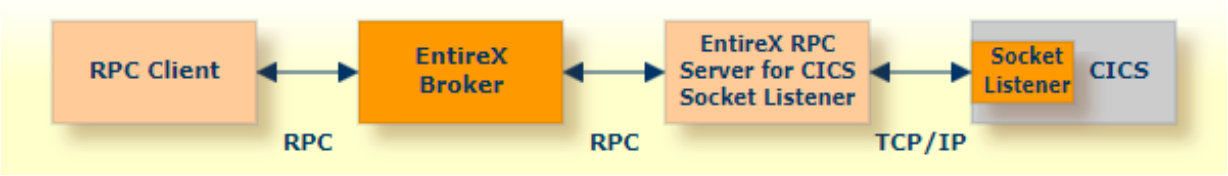
Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to the RPC Server for CICS Socket Listener

The EntireX RPC Server for CICS Socket Listener allows standard RPC clients to communicate with CICS programs running on IBM CICS®. All CICS interface types are supported: (DFHCOM-MAREA, Channel Container and Large Buffer).

Overview

The RPC Server for CICS Socket Listener acts on one side as an RPC server and on the other side as a client for CICS. The RPC Server for CICS Socket Listener is a Java-based component that can run on a different host to the one where CICS is running. This allows it to operate with a minimal footprint of EntireX on the CICS host. For details see *Preparing for CICS Socket Listener*. No configuration in CICS is required.



For local extraction, all source files have to be stored locally on the same machine where the Designer is running.

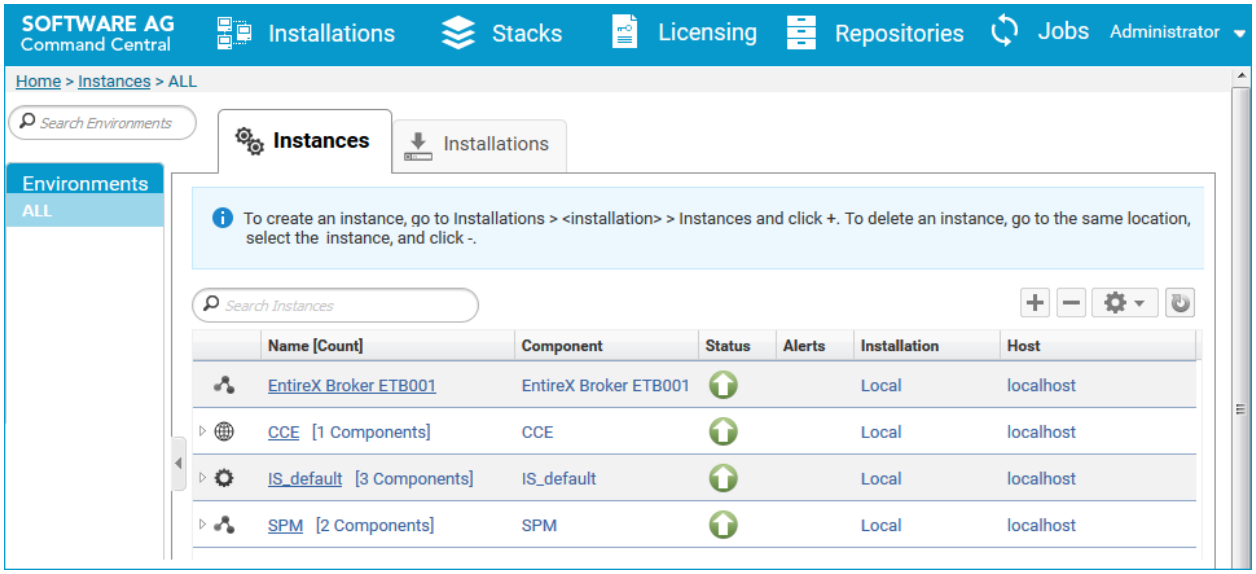
- For existing CICS COBOL programs, use the *Software AG IDL Extractor for COBOL* to extract the *Software AG IDL File* in the IDL Editor documentation for the RPC clients.
- For existing CICS PL/I programs, use the *Software AG IDL Extractor for PL/I* to extract the *Software AG IDL File* in the IDL Editor documentation for the RPC clients.

Remote extraction requires an RPC server running under z/OS with Extractor Service (Batch | IMS).

- For COBOL, see *Step 2: Select a COBOL Extractor Environment or Create a New One* in the IDL Extractor for COBOL documentation.
- For PL/I, see *Extract Software AG IDL File from a Remote PL/I RPC Environment* in the IDL Extractor for PL/I 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).



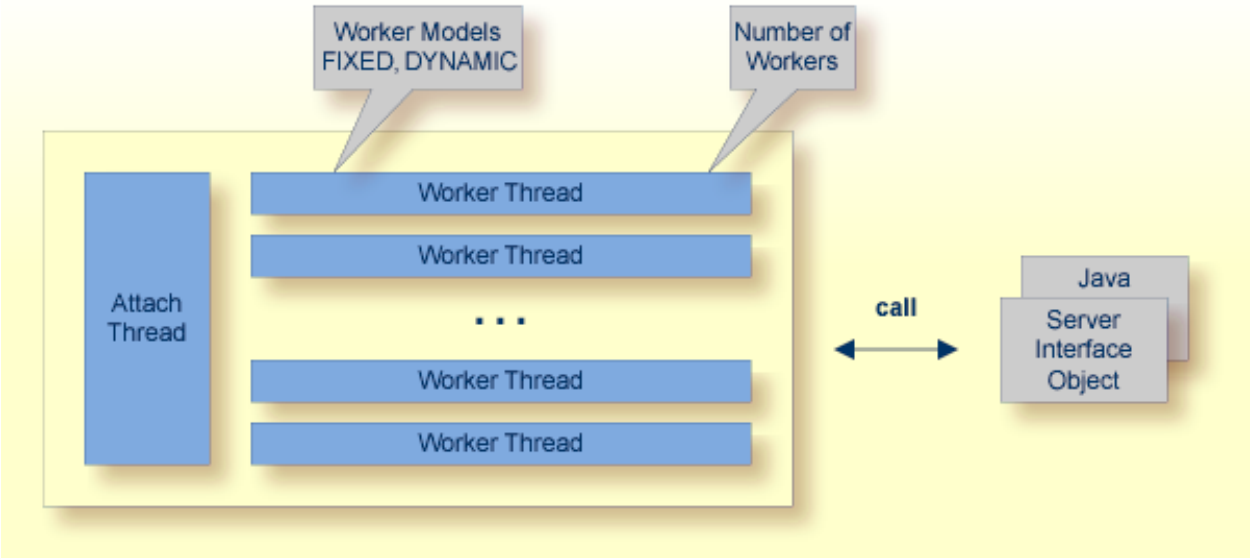
Command Central can assist with the following configuration, management, and monitoring tasks:

- 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 CICS Socket Listener using the Command Central GUI*. For the command-line interface, see *Administering the RPC Server for CICS Socket Listener 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 CICS Socket Listener 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 CICS Socket Listener*.

3 Administering the RPC Server for CICS Socket Listener using the Command Central GUI

This chapter describes how to administer the EntireX RPC Server for CICS Socket Listener, using the Command Central graphical user interface.

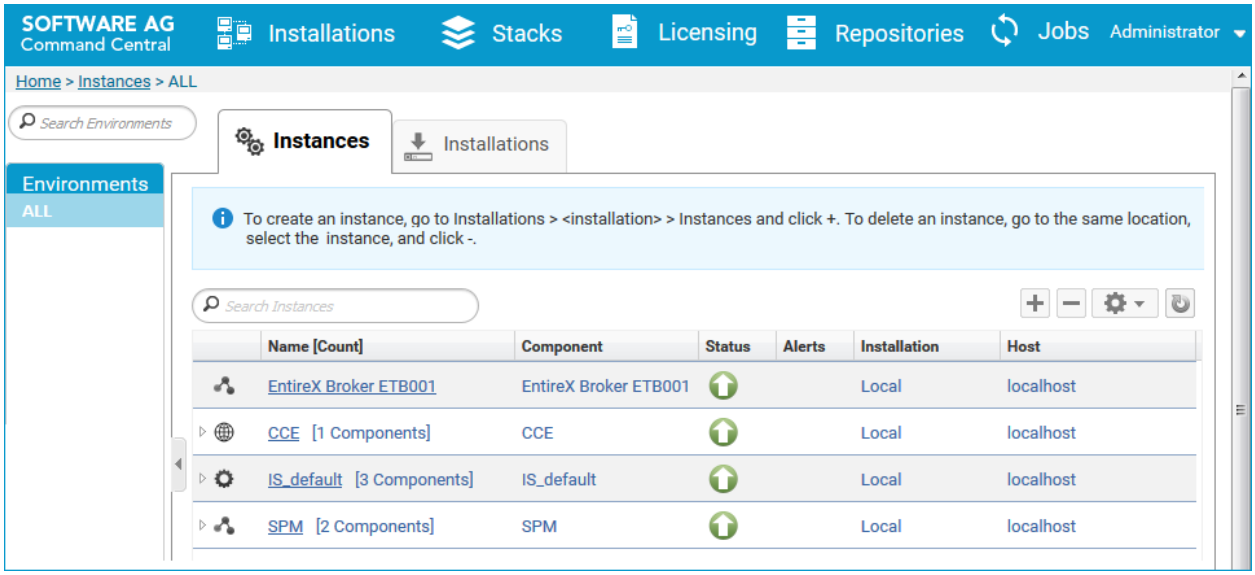
See also *Administering the RPC Server for CICS Socket Listener using the Command Central Command Line*. The core Command Central documentation is provided separately and is also available under **Guides for Tools Shared by Software AG Products** on the Software AG documentation website.

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.

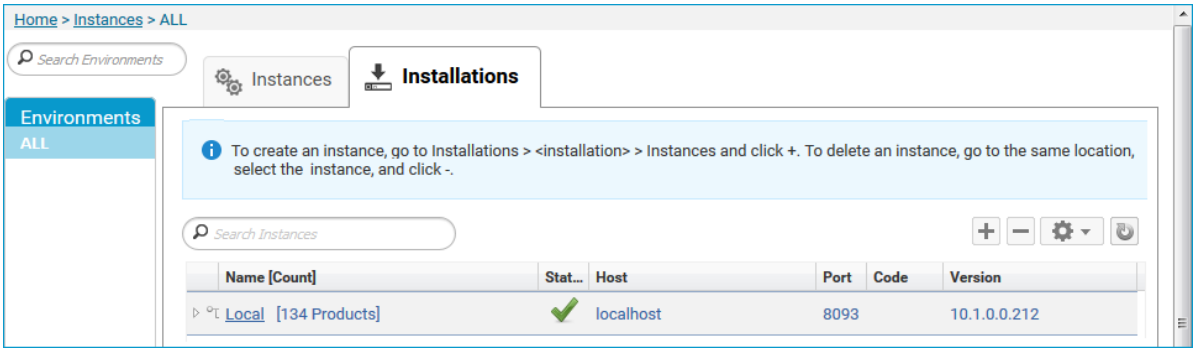
Provide your user credentials in the **Login** page and click **Log In**. This takes you to the page **Home > Instances**:



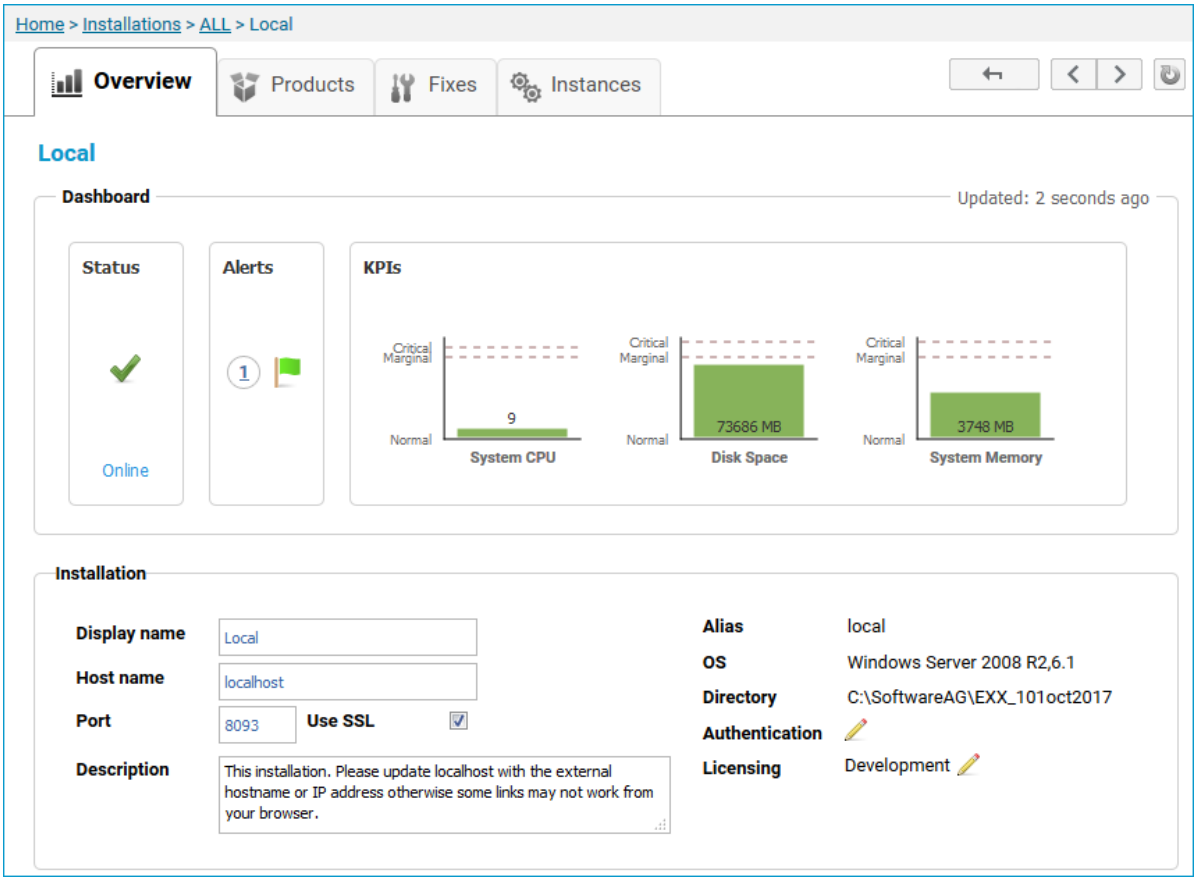
Creating an RPC Server Instance

To create an RPC Server for CICS Socket Listener instance

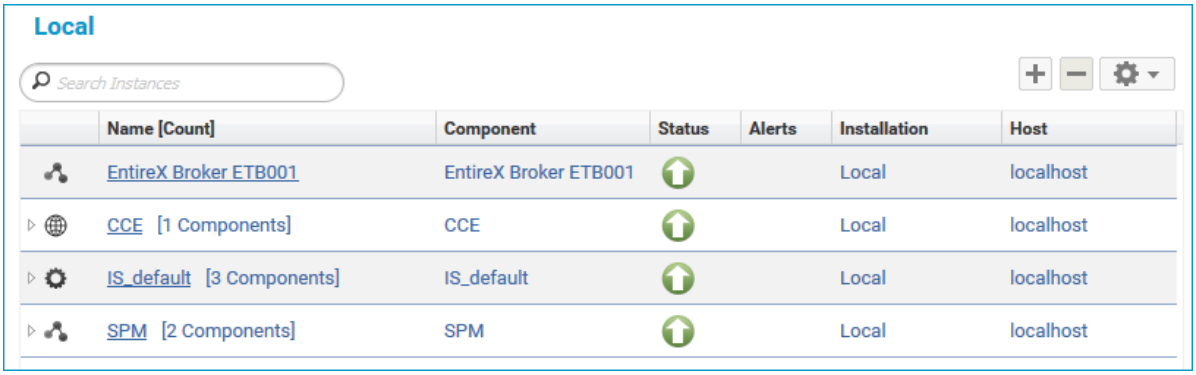
1 In the Command Central home page, click the **Installations** tab.



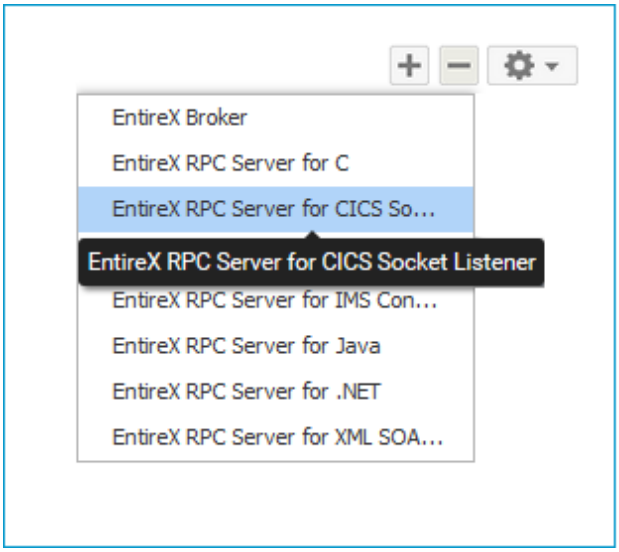
2 Click on the desired installation, for example **Local**, where you want to add an RPC Server for CICS Socket Listener instance.



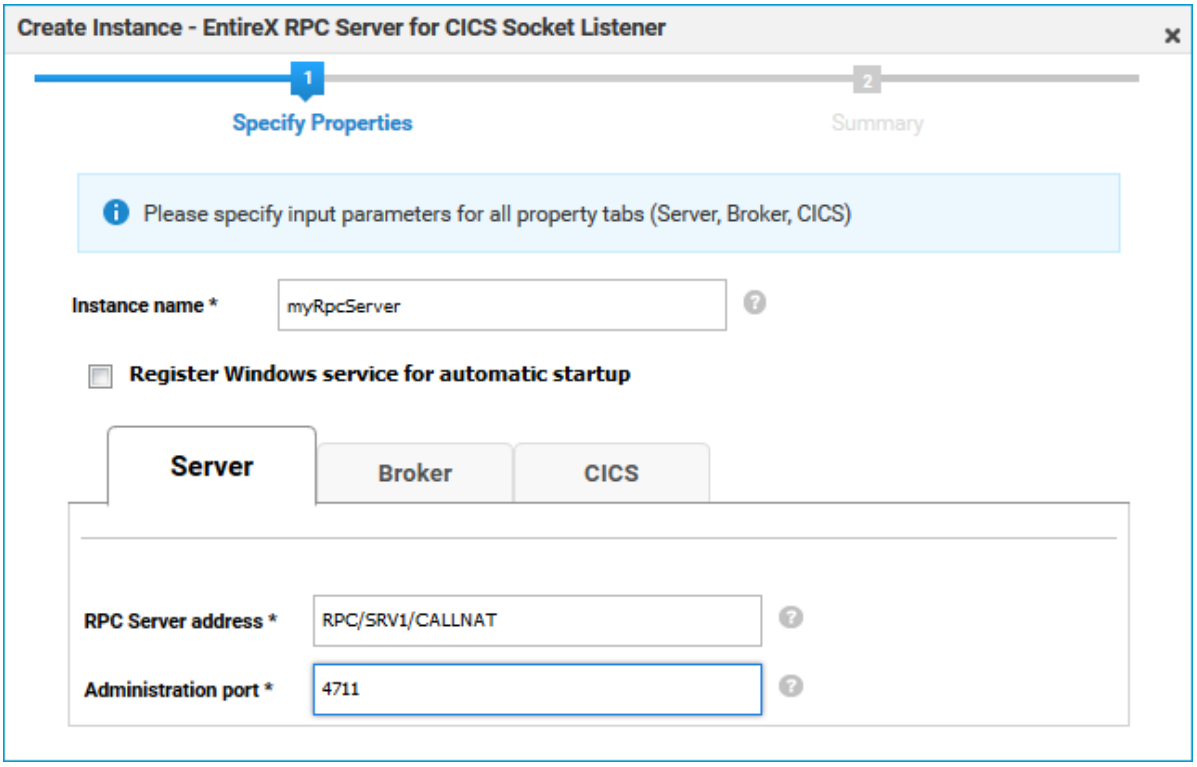
3 Click the **Instances** tab.



4 Click the button in the upper right corner above the list and choose **EntireX RPC Server for CICS Socket Listener**.



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

CICS Tab

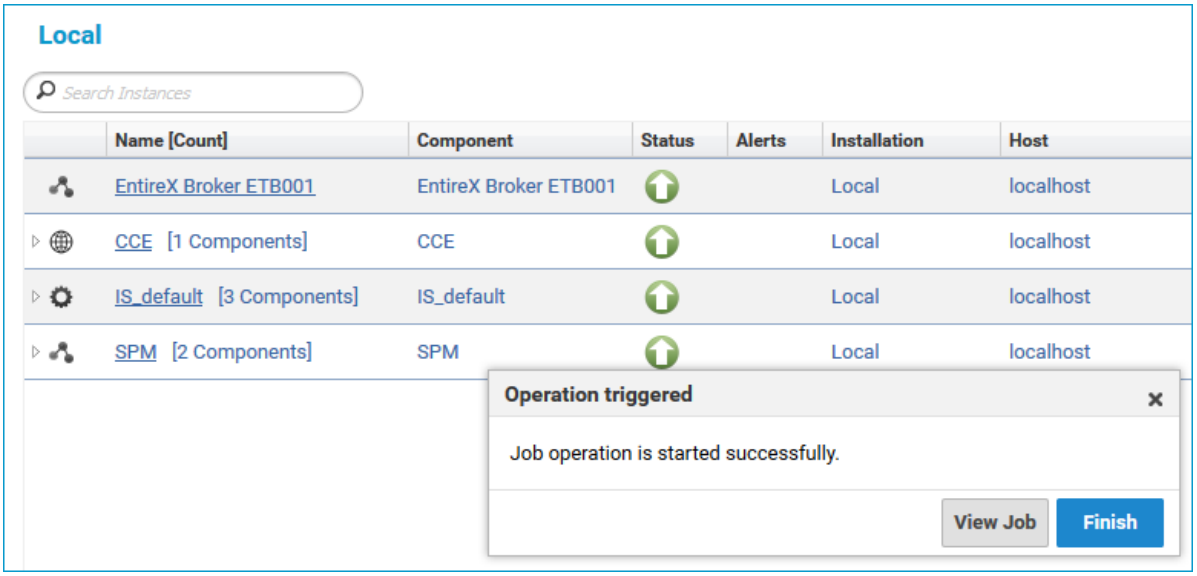
Here you can modify the CICS Socket Listener specific parameters.

Parameter	Description
Connection	
Transport	Required. Use TCP or SSL to communicate with CICS Socket Listener.
CICS host	Required. Host name or IP address where the CICS Socket Listener is running. See *Using the Broker ID in Applications* in the RPC Programming documentation.
CICS port	Required. TCP or SSL port number (1-65535) of the CICS Socket Listener.
CICS transaction ID	Required. Transaction ID (1-4 characters) defined for the RPC CICS RFE. Default is `XRFE`.
CICS encoding	Required. Specify the appropriate EBCDIC encoding used by your CICS installation. Default is codepage `cp037` with full Latin-1 character set.
Credentials	
CICS user	Optional. The user ID (max. 8 characters) for access to CICS as defined in your underlying mainframe security system (e.g. RACF).
CICS password	Optional. Password (max. 8 characters) as defined in your underlying mainframe security system (e.g. RACF).
Use pass ticket	Optional. Use pass ticket instead of password. See note.
Application name	Optional. Required if pass ticket is to be used instead of a password. Application name (1-8 characters) as defined in your underlying mainframe security system (e.g. RACF). See note.
Secured signon key	Optional. Required if pass ticket is to be used instead of a password. Secured signon key as defined in your underlying mainframe security system. Must be exactly 16 characters long. See note.

Note: PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See *Preparing for CICS Socket Listener* and *EntireX CICS Socket Listener* in the z/OS Installation documentation.

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

7 Press **Finish**.

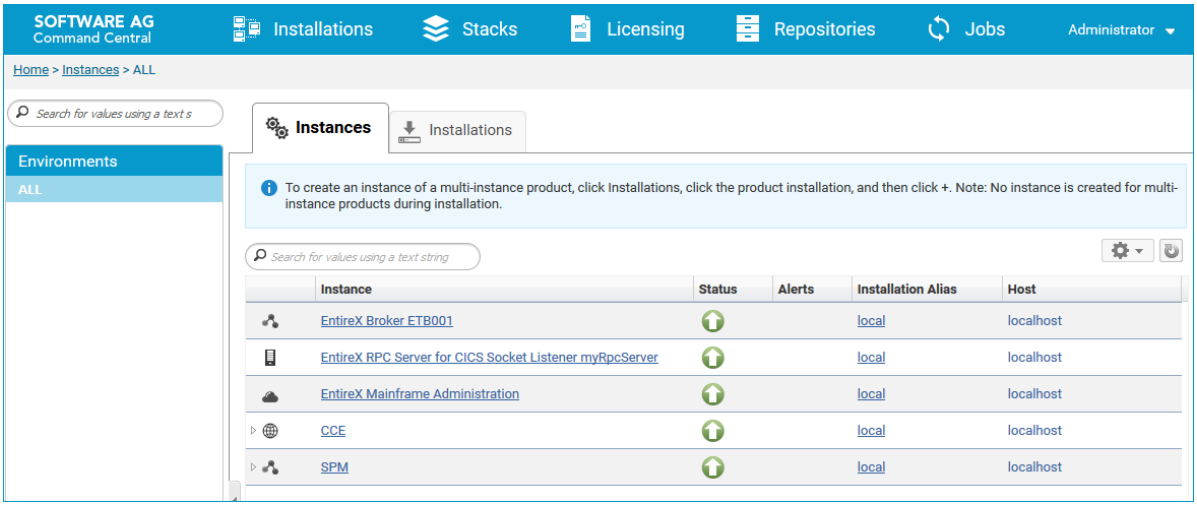


The new instance *myRpcServer* appears in the list.

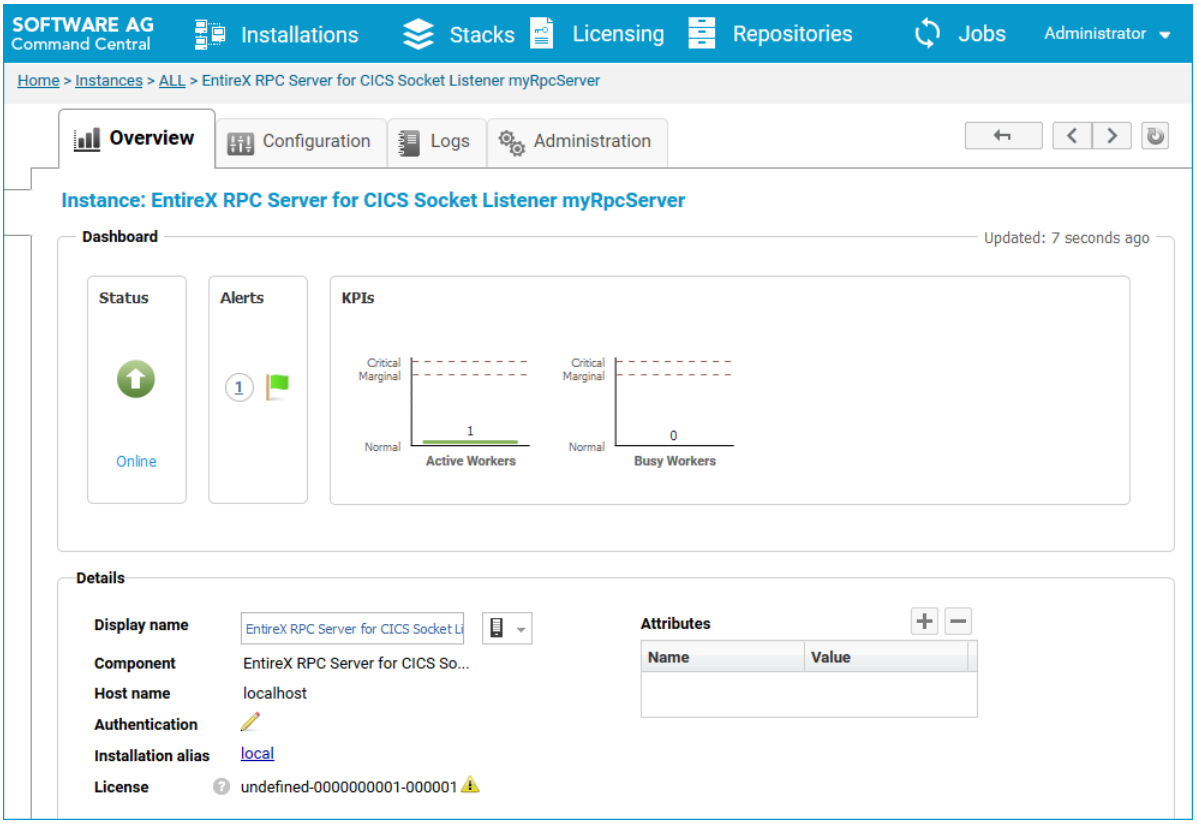
Configuring an RPC Server Instance

To configure an RPC Server for CICS Socket Listener instance

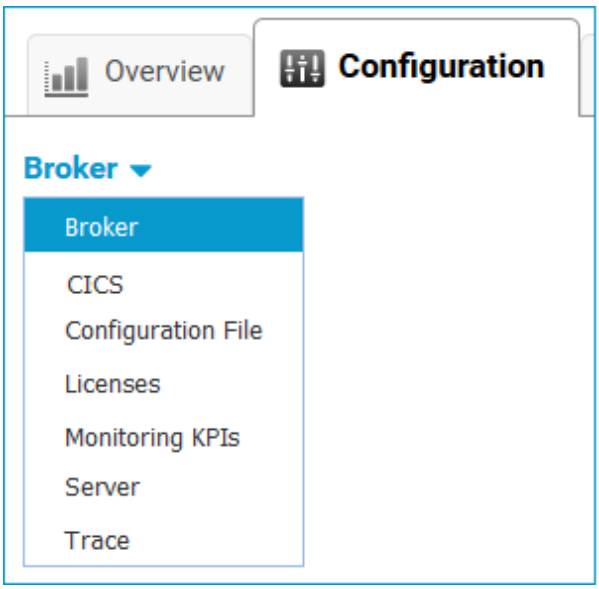
1 In the Command Central home page, click the **Instances** tab.



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**
- **CICS**
- **Configuration File**
- **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.

CICS

Here you can modify the CICS Socket Listener specific parameters. As a prerequisite, the CICS Socket Listener must be installed. See *Preparing for CICS Socket Listener*.

Parameter	Description
Connection	
Transport	Required. Use TCP or SSL to communicate with CICS Socket Listener.
CICS host	Required. Host name or IP address where the CICS Socket Listener is running. See *Using the Broker ID in Applications* in the RPC Programming documentation.
CICS port	Required. TCP or SSL port number (1-65535) of the CICS Socket Listener.
CICS transaction ID	Required. Transaction ID (1-4 characters) defined for the RPC CICS RFE. Default is `XRFE`.
CICS encoding	Required. Specify the appropriate EBCDIC encoding used by your CICS installation. Default is codepage `cp037` with full Latin-1 character set.
CICS SSL trust store	Optional. Specifies the location of the SSL trust store.

Parameter	Description
CICS SSL verify server	Optional. The RPC server as SSL client checks the identity of CICS Socket Listener as SSL server.
CICS socket timeout	Optional. Timeout (in seconds) for the CICS Socket Listener as SSL server. Default is 20 seconds.
Credentials	
CICS user	Optional. The user ID (max. 8 characters) for access to CICS as defined in your underlying mainframe security system (e.g. RACF).
CICS password	Optional. Password (max. 8 characters) as defined in your underlying mainframe security system (e.g. RACF).
Use pass ticket	Optional. Use pass ticket instead of password. See note.
Application name	Optional. Required if pass ticket is to be used instead of a password. Application name (1-8 characters) as defined in your underlying mainframe security system (e.g. RACF). See note.
Secured signon key	Optional. Required if pass ticket is to be used instead of a password. Secured signon key as defined in your underlying mainframe security system. Must be exactly 16 characters long. See note.

Note: PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See ***Preparing for CICS Socket Listener*** and *EntireX CICS Socket Listener* in the z/OS Installation documentation.

Configuration File

Here you can view/edit the configuration file of the RPC Server for CICS Socket Listener.

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 CICS Socket Listener: Active Workers and Busy Workers.

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for CICS Socket Listener. 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 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 CICS Socket Listener.

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.

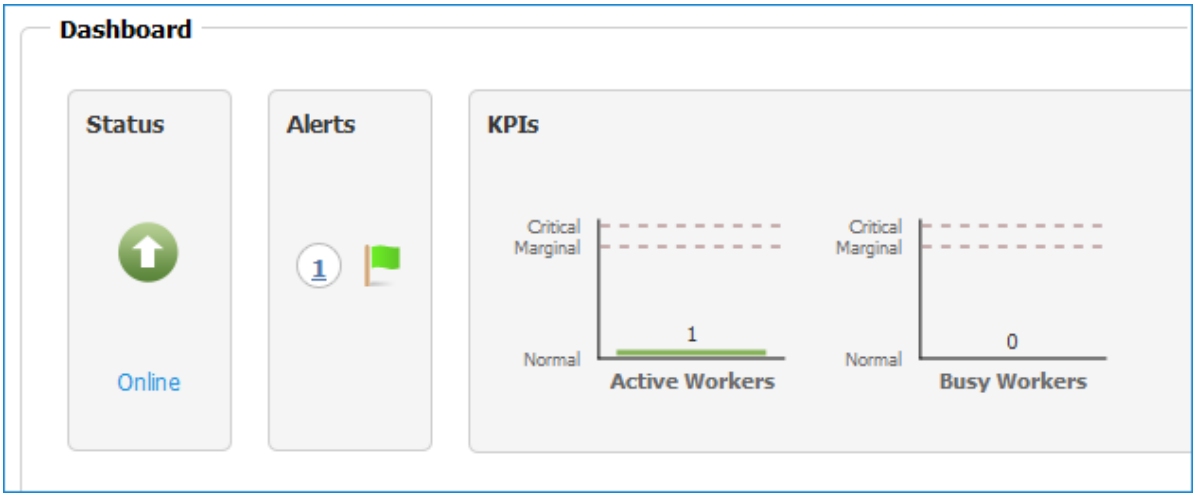
4 Click **Edit** to modify the parameters on your selected configuration type.

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



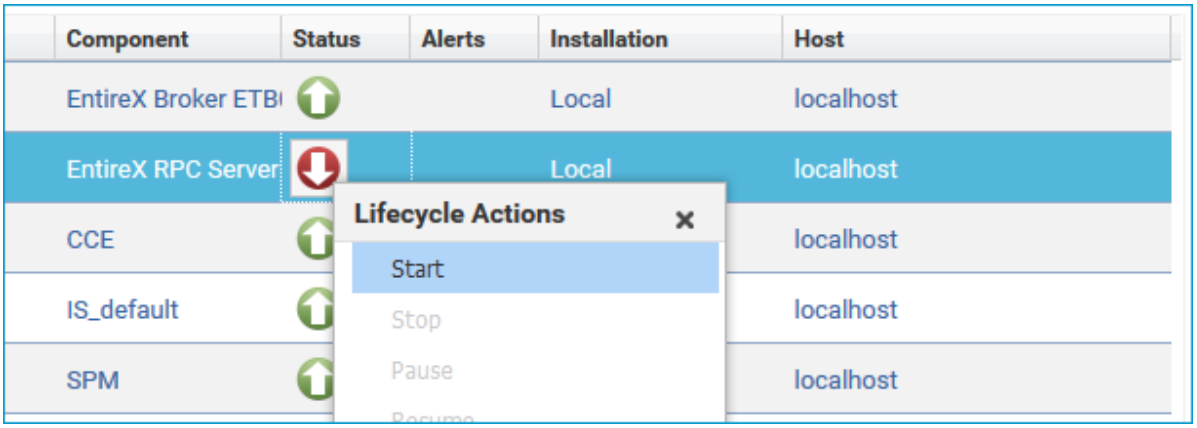
The visual key performance indicators (KPIs) and alerts enable you to monitor the RPC Server for CICS Socket Listener's health.

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 CICS Socket Listener instance from the Instances tab

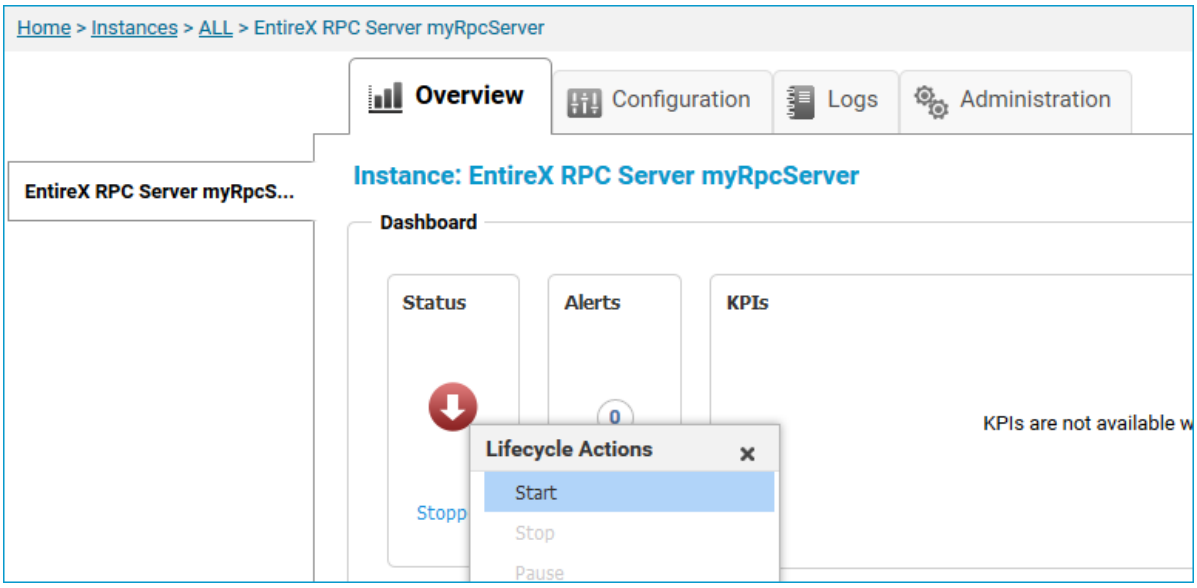
1 In the Command Central home page, click the **Instances** tab.



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

To start an RPC Server for CICS Socket Listener instance from its Overview tab

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

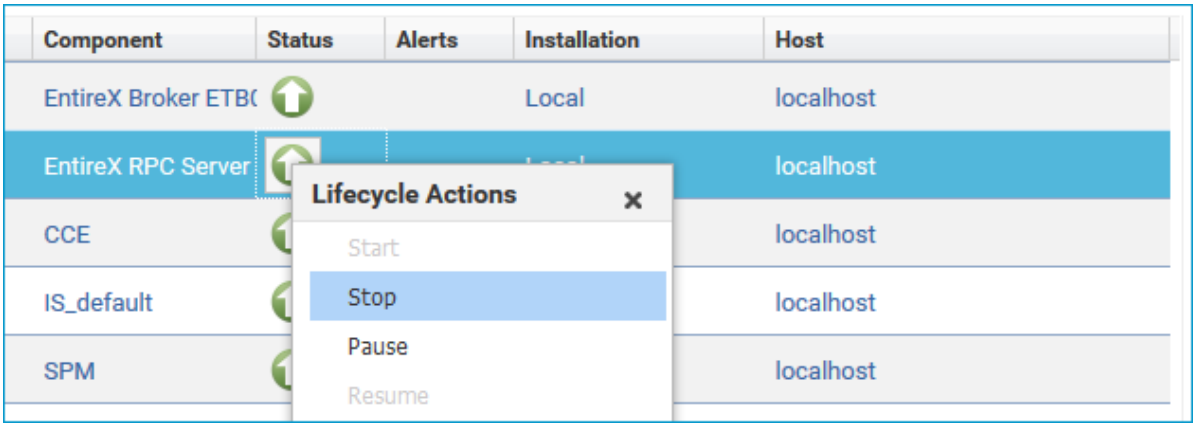


2 Select the status, and from the context menu choose **Start**.

Stopping an RPC Server Instance

To stop an RPC Server for CICS Socket Listener instance from the Instances tab

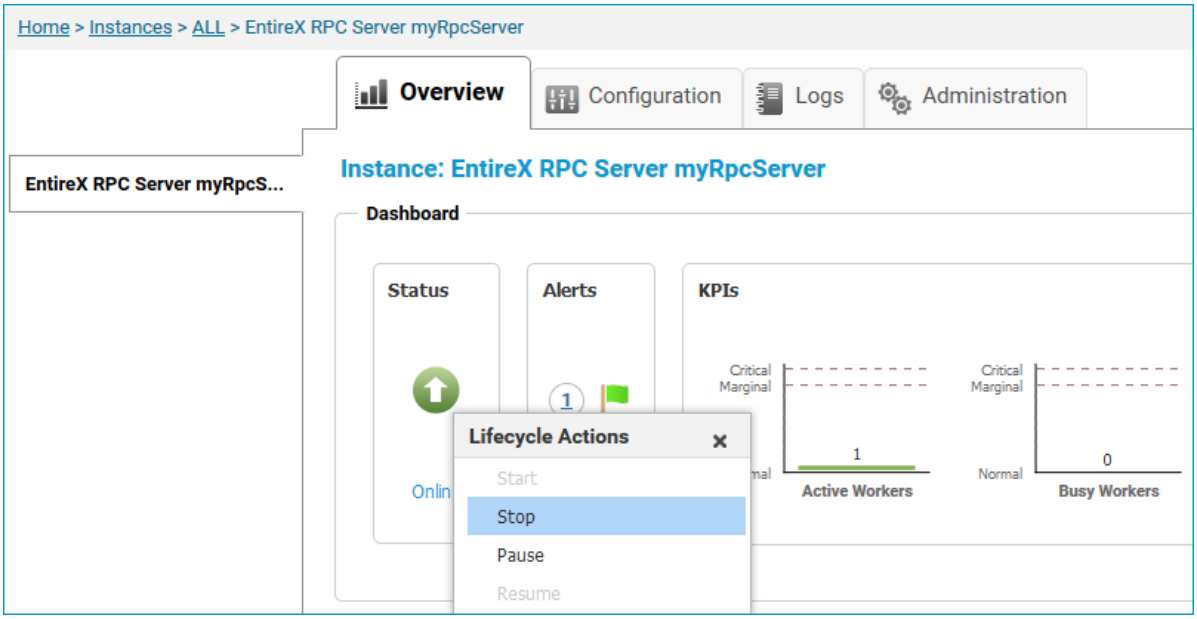
1 In the Command Central home page, click the **Instances** tab.



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

To stop an RPC Server for CICS Socket Listener instance from its Overview tab

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



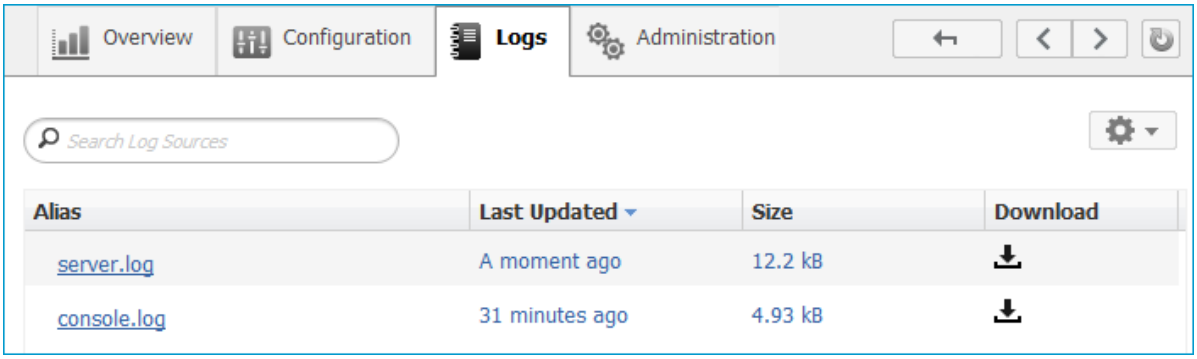
2 Select the status, and from the context menu choose **Stop**.

Inspecting the Log Files

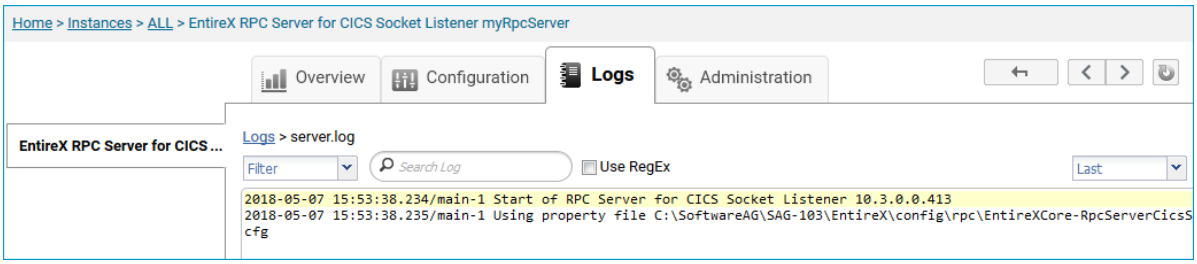
To inspect the log files of an RPC Server for CICS Socket Listener instance

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

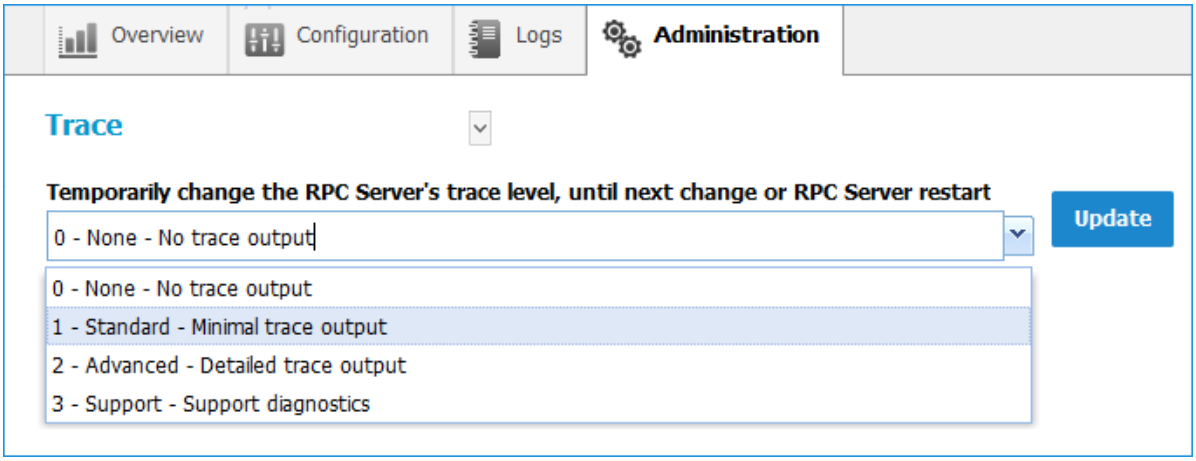


Changing the Trace Level Temporarily

To temporarily change the trace level of an RPC Server for CICS Socket Listener instance

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

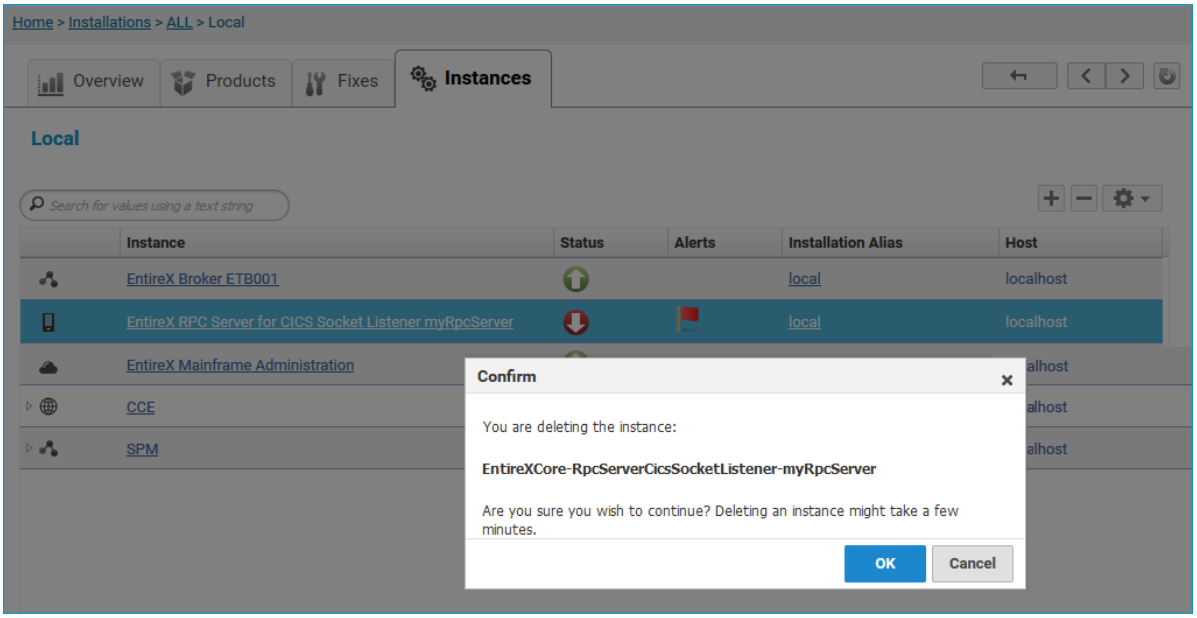


Note: If you want to set the trace level permanently, see *Trace Level* under *Configuring an RPC Server Instance*.

Deleting an RPC Server Instance

To delete an RPC Server for CICS Socket Listener instance

1 In the list of EntireX RPC Server for CICS Socket Listener 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 CICS Socket Listener instance.

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

4 Administering the RPC Server for CICS Socket Listener using the Command Central Command Line

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

Administering the RPC Server for CICS Socket Listener using the Command Central GUI is described under *Administering the RPC Server for CICS Socket Listener 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`	`RpcServerCicsSocketListener`	Required. EntireXCore instance type of RPC server. Must be `"RpcServerCicsSocketListener"`.
	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.

Command	Parameter	Value	Description
	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.
	cics.sl.transport	ssl | tcp	Required. Use TCP or SSL to communicate with CICS Socket Listener.
	cics.sl.host	*name*	Required. Host name or IP address where the CICS Socket Listener is running. See *Using the Broker ID in Applications* in the RPC Programming documentation.
	cics.sl.port	1-65535	Required. TCP or SSL port number (1-65535) of the CICS Socket Listener.
	cics.sl.transaction	XRFE | *transaction*	Required. Transaction ID (1-4 characters) defined for the RPC CICS RFE. Default is XRFE.
	cics.sl.encoding	cp037 | *codepage*	Required. Specify the appropriate EBCDIC encoding used by your CICS installation. Default is codepage cp037 with full Latin-1 character set.
	cics.sl.user	*user*	Optional. The user ID (max. 8 characters) for access to CICS as defined in your underlying mainframe security system (e.g. RACF).
	cics.sl.password	*password*	Optional. Password (max. 8 characters) as defined in your underlying mainframe security system (e.g. RACF).
	pass.ticket	true | false	Optional. Use pass ticket instead of password. See note.
	cics.sl.application.name	*application_name*	Optional. Required if pass ticket is to be used instead of a password. Application name (1-8 characters) as defined in your underlying mainframe security system (e.g. RACF). See note.
	cics.sl.secured.signonkey	*key_name*	Optional. Required if pass ticket is to be used instead of a password. Secured signon key as defined in your underlying mainframe security system. Must be exactly 16 characters long. See note.

Note: PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See *Preparing for CICS Socket Listener* and *EntireX CICS Socket Listener* in the z/OS Installation documentation.

Example

- To create a new instance for an installed EntireX of the type "RpcServerCicsSocketListener", with name "MyRpcServer", with server address "RPC/SRV1/CALLNAT", using administration port 5757, with broker host name "localhost", listening on broker port 1971, transmitting CICS request with the transport over "tcp", to host "cicsHost", via port "5822", with transaction ID "XRFE" and encoding "cp037", in the installation with alias name "local":

```
sagcc create instances local EntireXCore type=RpcServerCicsSocketListener
instance.name=MyRpcServer server.address=RPC/SRV1/CALLNAT server.adminport=5757
broker.host=localhost broker.port=1971 cics.sl.transport=TCP cics.sl.host=cicsHost
cics.sl.port=5822 cics.sl.transaction=XRFE cics.sl.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 CICS Socket Listener. Any changes to parameters will be used the next time you start the RPC server.

- Broker
- CICS
- Configuration File
- Monitoring KPIs
- Server
- Trace Level

Broker

Here you can administer the parameters used for communication between the RPC Server for CICS Socket Listener 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-RpcServerCicsSocketListener-".
	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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-"`.
	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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local" from the file *broker.json* in the current working directory:

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

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

CICS

Here you can modify the CICS-specific configuration parameters. As a prerequisite, the CICS Socket Listener must be installed. See ***Preparing for CICS Socket Listener***.

Parameters

Parameter	Value	Description
`CicsSocketListenerTransport`	`ssl` | `tcp`	Required. Use TCP or SSL to communicate with CICS Socket Listener.
`CicsSocketListenerHost`	*`name`*	Required. Host name or IP address where the CICS Socket Listener is running. See *Using the Broker ID in Applications* in the RPC Programming documentation.
`CicsSocketListenerPort`	`1-65535`	Required. TCP or SSL port number (1-65535) of the CICS Socket Listener.
`CicsSocketListenerTransaction`	`XRFE` | *`transaction`*	Required. Transaction ID (1-4 characters) defined for the RPC CICS RFE. Default is `XRFE`.
`CicsSocketListenerEncoding`	`cp037` | *`codepage`*	Required. Specify the appropriate EBCDIC encoding used by your CICS installation. Default is codepage `cp037` with full Latin-1 character set.
`CicsSocketListenerSslTrustStore`	*`file_path`*	Optional. Specifies the location of the SSL trust store.
`CicsSocketListenerVerifyServer`	`true` | `false`	Optional. The RPC server as SSL client checks the identity of CICS Socket Listener as SSL server.
`CicsSocketListenerTimeout`	`20` | *`n`*	Optional. Timeout (in seconds) for the CICS Socket Listener as SSL server. Default is 20 seconds.
`CicsSocketListenerUser`	*`user`*	Optional. The user ID (max. 8 characters) for access to CICS as defined in your underlying mainframe security system (e.g. RACF).
`CicsSocketListenerPassword`	*`password`*	Optional. Password (max. 8 characters) as defined in your underlying mainframe security system (e.g. RACF).

Parameter	Value	Description
`CicsSocketListenerPassTicket`	`true | false`	Optional. Use pass ticket instead of password. See note.
`CicsSocketListenerApplicationName`	*`application_name`*	Optional. Required if pass ticket is to be used instead of a password. Application name (1-8 characters) as defined in your underlying mainframe security system (e.g. RACF). See note.
`CicsSocketListenerSecuredSignonKey`	*`key_name`*	Optional. Required if pass ticket is to be used instead of a password. Secured signon key as defined in your underlying mainframe security system. Must be exactly 16 characters long. See note.

Note: PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See ***Preparing for CICS Socket Listener*** and *EntireX CICS Socket Listener* in the z/OS Installation documentation.

Displaying the CICS Socket Listener Specific 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-RpcServerCicsSocketListener-"`.
	`instanceid`	Required. Must be `"CICS_SOCKET_LISTENER"`.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Example 1

- To display the CICS Socket Listener specific parameters of the RPC Server for CICS Socket Listener "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-MyRpcServer CICS_SOCKET_LISTENER
```

Example 2

- To store the CICS Socket Listener specific parameters in the file *cics.json* in the current working directory:

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-MyRpcServer CICS_SOCKET_LISTENER -o
cics.json
```

Resulting output file in JSON format:

```
{
    "CicsSocketListenerTransport": "TCP",
    "CicsSocketListenerHost": "ibm2",
    "CicsSocketListenerPort": "1234",
    "CicsSocketListenerEncoding": "cp037",
    "CicsSocketListenerSslTrustStore": "",
    "CicsSocketListenerSslVerifyServer": "true",
    "CicsSocketListenerTimeout": "20",
    "CicsSocketListenerUser": "",
    "CicsSocketListenerPassTicket": "",
    "CicsSocketListenerApplicationName": "",
    "CicsSocketListenerSecuredSignonKey": "",
    "CicsSocketListenerTransaction": "XRFE"
    }
```

Updating the CICS Socket Listener Specific 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-RpcServerCicsSocketListener-"`.
	`instanceid`	Required. Must be `"CICS_SOCKET_LISTENER"`.
	`-i` *`file`*	Optional. Specifies the file from where you want the input read.

Example

- To modify the CICS Socket Listener parameters, get the file *cics.json* with the `get` command. Edit the parameters in this file, and update the RPC Server for CICS Socket Listener "MyRpc-Server" in the installation with alias name "local" with the following command:

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-MyRpcServer CICS_SOCKET_LISTENER -i
cics.json
```

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

Configuration File

Here you can administer the configuration file of the RPC Server for CICS Socket Listener. Any changes will take effect after the next restart.

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-RpcServerCicsSocketListener-".
	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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local":

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-"`.
	`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-RpcServerCicsSocketListener-MyRpcServer CONFIGURATION -i
configuration.json
```

Monitoring KPIs

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

Parameters

Key performance indicators (KPIs) enable you to monitor the health of your RPC Server for CICS Socket Listener. 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-RpcServerCicsSocketListener-"`.
	`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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-`".
	`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-RpcServerCicsSocketListener-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 CICS Socket Listener.

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 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-RpcServerCicsSocketListener-"`.
	`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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-"`.
	`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-RpcServerCicsSocketListener-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 CICS Socket Listener.

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-RpcServerCicsSocketListener-"`.
	`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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local" on stdout:

```
sagcc get configuration data local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-".
	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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-"`.

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-RpcServerCicsSocketListener-"`.

Example

- To display state information about the RPC Server for CICS Socket Listener:

```
sagcc get monitoring state local
EntireXCore-RpcServerCicsSocketListener-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 CICS Socket Listener, 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-RpcServerCicsSocketListener-"`.

Example

- To start the RPC Server for CICS Socket Listener "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle start local
EntireXCore-RpcServerCicsSocketListener-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 CICS Socket Listener, 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-RpcServerCicsSocketListener-"`.

Example

- To stop the RPC Server for CICS Socket Listener "MyRpcServer" in the installation with alias name "local":

```
sagcc exec lifecycle stop local EntireXCore-RpcServerCicsSocketListener-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 CICS Socket Listener. 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

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-RpcServerCicsSocketListener-"`.

Example

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

```
sagcc list diagnostics logs local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-"`.
	`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-RpcServerCicsSocketListener-MyRpcServer server.log tail
```

Example 2

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

```
sagcc get diagnostics logs local
EntireXCore-RpcServerCicsSocketListener-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 CICS Socket Listener "MyRpcServer" in the installation with alias name "local" in JSON format on stdout:

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

Example 2

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

```
sagcc exec administration component local
EntireXCore-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-`".
	`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-RpcServerCicsSocketListener-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-RpcServerCicsSocketListener-`".

Example

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

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

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

5 Administering the RPC Server for CICS Socket Listener

The EntireX RPC Server for CICS Socket Listener allows standard RPC clients to communicate with CICS programs running on IBM CICS®. All CICS interface types are supported: (DFHCOM-MAREA, Channel Container and Large Buffer).

Customizing the RPC Server

The following are used to set up the RPC Server for CICS Socket Listener:

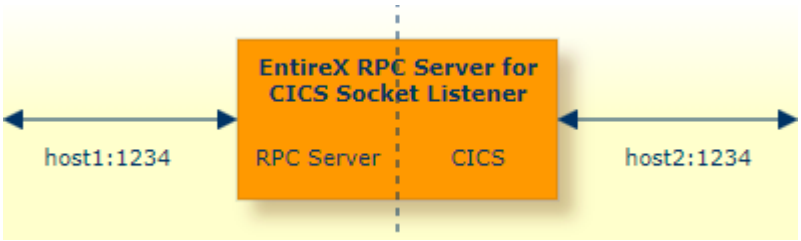
- Configuration File
- Start Script

Configuration File

The default name of the configuration file is *cicssocketlistener.properties*. The RPC Server for CICS Socket Listener 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 CICS Socket Listener.



Configuring more than one RPC Server

If you configure more than one RPC Server for CICS Socket Listener that connect 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 CICS Socket Listener is called *cicssocketlistenerserver.bsh* (UNIX) or *cicssocketlistenerserver.bat* (Windows) and is provided in the *bin* folder of the installation directory. You may customize this file.

Configuring the RPC Server Side

The RPC Server for CICS Socket Listener 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`	`CICSSLRPCServer`	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 CICS Socket Listener 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.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, default is standard output.
entirex.trace	-trace	0	Trace level (1,2,3).

Configuring the CICS Socket Listener Side

These properties are used to configure the connection to CICS. As a prerequisite, the CICS Socket Listener must be installed. See ***Preparing for CICS Socket Listener***.

Alternatively, 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.

Name	Default Value	Explanation
`cics.sl.host`		Required. Host name or IP address where the CICS Socket Listener is running. See *Using the Broker ID in Applications* in the RPC Programming documentation.
`cics.sl.port`		Required. TCP or SSL port number (1-65535) of the CICS Socket Listener.
`cics.sl.transaction`	`XRFE`	Required. Transaction ID (1-4 characters) defined for the RPC CICS RFE. Default is `XRFE`.
`entirex.bridge.targetencoding`	`cp037`	Required. Specify the appropriate EBCDIC encoding used by your CICS installation. Default is codepage `cp037` with full Latin-1 character set.
`cics.sl.sockettimeout`	`20`	Optional. Timeout (in seconds) for the CICS Socket Listener as SSL server. Default is 20 seconds.
`cics.sl.userid`		Optional. The user ID (max. 8 characters) for access to CICS as defined in your underlying mainframe security system (e.g. RACF).
`cics.sl.password`		Optional. Password (max. 8 characters) as defined in your underlying mainframe security system (e.g. RACF).
`cics.sl.sslparams`		SSL parameters (optional). Same syntax as Broker ID.
`cics.sl.application.name`		Optional. Required if pass ticket is to be used instead of a password. Application name (1-8 characters) as defined in your underlying mainframe security system (e.g. RACF). See note. This property is ignored if `cics.sl.password` is set.
`cics.sl.secured.signonkey`		Optional. Required if pass ticket is to be used instead of a password. Secured signon key as defined in your underlying mainframe security system. Must be exactly 16 characters long. See note. This property is ignored if `cics.sl.password` is set.

Name	Default Value	Explanation
`cics.sl.user.transaction.id`		Optional. The CICS transaction identifier (max. 4 characters) that will be used to run the CICS program in a separate user transaction.

Note: PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See ***Preparing for CICS Socket Listener*** and *EntireX CICS Socket Listener* in the z/OS Installation documentation.

Using SSL/TLS with the RPC Server

To use SSL with the RPC Server for CICS Socket Listener, you need to configure two sides:

- **CICS Side**
  See parameter *`cics.sl.sslparams`*.
- **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 CICS Socket Listener 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 CICS Socket Listener

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

Stopping the RPC Server

To stop the RPC Server for CICS Socket Listener

- 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 CICS Socket Listener

- 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 CICS Socket Listener as a Windows Service

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

   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 *cicssocketlistenerserver.bat*, the command will be

```
RPCService -install -ext MyServer ↩
-script install_path\EntireX\bin\cicssocketlistenerserver.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 CICS Socket Listener to the Broker. It is visible with Broker Command and Information 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.

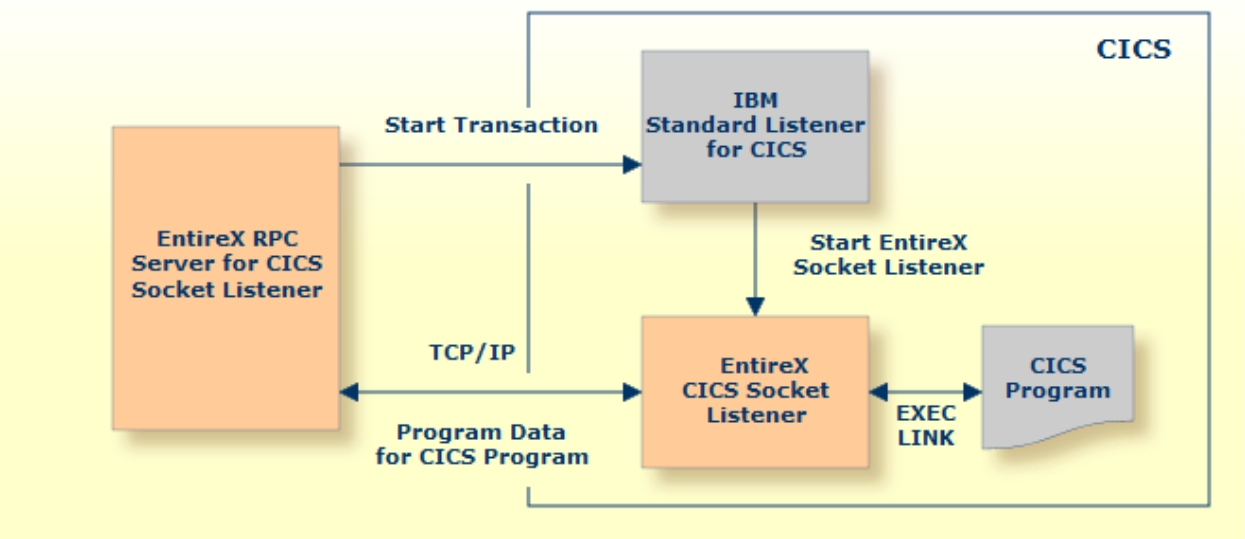
For the RPC Server for CICS Socket Listener, these values are:

Identification Part	Value
Application name	`ANAME=RPC Server for CICS Socket Listener`
Node name	`ANODE=<host name>`
Application type	`ATYPE=Java`
Version	`AVERS=10.5.0.0`

6 Preparing for CICS Socket Listener

Overview

The CICS Socket Listener is used by the RPC Server for CICS Socket Listener. Apart from installation there is no configuration necessary in CICS. Configuration is done with the RPC Server for CICS Socket Listener. See *Configuring an RPC Server Instance* > *CICS* using the Command Central GUI | Command Line.



The implementation for CICS is based on the CICS standard listener provided by IBM. With this listener you can launch CICS transaction via TCP/IP. The launched transaction takes the TCP connection and continues the communication with the launching process.

Depending on your platform, more information on configuring the IBM standard listener for CICS can be found under the following IBM documentation:

- *z/OS Communications Server: IP CICS Socket Guide*
- *z/VSE TCP/IP Support*

Installing the CICS Socket Listener

The CICS Socket Listener is installed

- together with the RPC Server for CICS, see *Installing the RPC Server for CICS* (z/OS | z/VSE), or
- separately, see *EntireX CICS Socket Listener* in the z/OS | z/VSE Installation documentation

Configuring the IBM Standard Listener

Depending on your platform, more information on configuring the IBM standard listener for CICS can be found under the following IBM documentation:

- *z/OS Communications Server: IP CICS Socket Guide*
- *z/VSE TCP/IP Support*

To start/stop the IBM standard listener

- Use the CICS supplied transaction EZAO. The listener is automatically started/stopped when CICS is started or stopped.

To configure the IBM standard listener

- Use the CICS-supplied transaction `EZAC,ALT,LISTENER`.
  - For `SECEXIT`, define `EXXRFECS`.
  - Make sure the `PORT` number of the IBM standard listener corresponds to the configuration parameter `CICS port`. See *Configuring an RPC Server Instance* > *CICS* using the Command Central GUI | Command Line.

User Transaction Support

User transaction support means that each CICS program runs in a separate CICS task (the user transaction). This separate CICS task is started exclusively for a CICS program. After the CICS program has been executed, the user transaction terminates. If the CICS program issues a CICS `ABEND`, the user transaction is terminated. If you are running multiple CICS programs within an RPC conversation, the user transaction remains active and processes all CICS programs belonging to the conversation.

To run the CICS programs in a separate user transaction

1 Define your user transaction ID in the RPC Server for CICS Socket Listener, for example:

```
cics.sl.user.transaction.id=UTSK
```

For more information see *Configuring the CICS Socket Listener Side*.

2 Create the CSD for the user transaction ID.

3 In this CSD definition, specify `EXXRFECU` for the `PROGRAM` attribute. To define, for example, `UTSK` as user transaction ID, use the following commands:

```
DEFINE TRANSACTION(UTSK) GROUP(...)
DESCRIPTION(CICS Socket Listener user transaction)
PROGRAM(EXXRFECU)
```