webMethods EntireX

Administration

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.

Document ID: EXX-ADMIN-105-20220422GENERAL

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 Environment Variables in EntireX

This chapter gives an overview of environment variables in EntireX and how they are used.

Table of Environment Variables

The table below provides an overview of environment variables used on the various platforms supported by EntireX.

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Windows	UNIX	z/VSE			
`EXXDIR`			x		R	Top level directory for EntireX.	
`EXXVERS`			x		R	Version level directory of the EntireX. Deprecated. Kept for reasons of compatibility with earlier versions.	
`PATH`			x		R	System variable. Additional program directories required by EntireX are added to this variable by the EntireX environment script.	See *Shell Environment Settings*.
`LD_LIBRARY_PATH`			x		R	System variable. Additional shared library directories required by EntireX are added to this variable by the EntireX environment script.	See *Shell Environment Settings*.
`LIBPATH`			x		R	Same as `LD_LIBRARY_PATH` on AIX.	See *Shell Environment Settings*.
`CLASSPATH`		x	x		R	System variable. Additional JAR file path entries required by EntireX are added to this variable by the EntireX environment script (UNIX) or during installation (Windows).	
`ETB_ATTR`		x	x		O	Value of Broker attribute file. Set automatically by the Broker startup shell script.	See *Broker Attributes*.
`ETB_LOG`		x	x		O	Accounting file.	See *Accounting in EntireX Broker*.
`ETB_NONACT` `NONACT`	x	x	x	x	O	Limits the TCP/IP connection lifetime.	Stub-to-broker connection non-activity time in seconds. If not 0, connections with a non-activity time greater than `ETB_NONACT` will be closed. See *Limiting the TCP/IP Connection Lifetime* in the platform-specific *Stub Administration* sections of the EntireX documentation.

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Windows	UNIX	z/VSE			
ETB_SOCKETPOOL	x	x	x		O	Values: ON (default) or OFF to establish an affinity between threads and TCP/IP connections in a DVIPA environment.	See *Support of Clustering in a High Availability Scenario* under *Administering Broker Stubs* in the platform-specific Administration documentation.
ETB_STUBLOG STUBLOG	x	x	x	x	O	Trace level for the EntireX Broker API.	See *Application Stublog File* in the UNIX Administration documentation | *Tracing for Broker Stubs* under z/OS | Windows | z/VSE.
ETB_STUBLOGPATH		x	x		O	Under UNIX and Windows, the directory where the log file is created if ETB_STUBLOG is used.	
ETB_TIMEOUT TIMEOUT	x	x	x	x	O	Stub transport timeout.	See *Setting the Timeout for the Transport Method* in the platform-specific broker stub Administration documentation.
ERX_TRACELEVEL		x	x		O	Sets the trace level for EntireX RPC Runtime.	Tracing for various EntireX components such as DCOM Wrapper, .NET Wrapper and C Wrapper. See *Tracing webMethods EntireX* in the platform-specific Administration documentation.
ETB_TRANSPORT TRANSPORT	x	x	x	x	O	Sets the default transport method for Broker stubs.	See *Transport Methods for Broker Stubs* in the platform-specific broker stub Administration documentation.
ADALNK		x	x		O	The Adabas module that is needed by the Broker kernel to access the Adabas persistent store.	See *Managing the Broker Persistent Store* in the platform-specific Administration documentation.
ETBLNK			x		R	Identifies the absolute path to the broker stubs library if EntireX Broker has been installed.	See *Broker Stubs*.
ERX_TRACEFILE		x	x		O	Sets the name of the trace file for EntireX RPC Runtime.	Tracing for various EntireX components such as DCOM Wrapper, .NET Wrapper and C Wrapper. See *Tracing webMethods EntireX* in the platform-specific Administration documentation.
ERX_ETBAPIVERS		x	x		O	Determines the Broker API version to use.	EntireX components such as DCOM Wrapper, .NET Wrapper and C Wrapper and the EntireX Broker are able to detect automatically the best API version to use (if no environment variable is defined or the value 0 is assigned). However, for backward compatibility to EntireX

	Platform						
Environment Variable	z/OS	Windows	UNIX	z/VSE	Opt/ Req	Description	More Information
							Broker, it might be necessary to set a preferred API Version for the Broker.
ERX_CODEPAGE		x	x		O	Override or set a code page identifier used for ICU conversion for RPC clients generated with the C Wrapper, DCOM Wrapper and .NET Wrapper.	For more information see *Using Internationalization* with the C Wrapper | DCOM Wrapper | .NET Wrapper.
MONITOR_BROKER_ OUTFILE			x		O	Specifies an alternative output file for EntireX command-line monitoring script monitor_broker_to_csv_file .bat.	The default output is written to *<drive>:\Users\user_id\documents\ SoftwareAG\EntireX\out_monitor_broker .csv*. See *Monitoring Broker* in the Monitoring EntireX with Command-line Scripts documentation.
MONITOR_CLIENT_ OUTFILE			x		O	Specifies an alternative output file for EntireX command-line monitoring script monitor_client_to_csv_file .bat.	The default output is written to *<drive>:\Users\user_id\documents\ SoftwareAG\EntireX\out_monitor_clients .csv*. See *Monitoring Clients*.
MONITOR_SERVICE_ OUTFILE			x		O	Specifies an alternative output file for EntireX command-line monitoring script monitor_service_to_csv_file .bat.	The default output is written to *<drive>:\Users\user_id\documents\ SoftwareAG\EntireX\out_monitor_service .csv*. See *Monitoring Services*.
MONITOR_VERIFY			x		O	If MONITOR_VERIFY=YES, an EntireX monitoring script that writes to a CSV file pauses on first execution so you can confirm that the correct parameters are being used. If MONITOR_VERIFY=NO, the monitoring script writes to CSV file without waiting for your confirmation.	
NA2_BKDBGS		x	x		O	Security exit debug level. Used for protecting the Broker kernel on UNIX and Windows to leverage the local security system.	
NA2_BKDBGF		x	x		O	Security exit debug file. Used for protecting the Broker kernel on UNIX and Windows to leverage the local security system.	See *Setting up EntireX Security for Broker Kernel* in the UNIX and Windows post-installation documentation.

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Windows	UNIX	z/VSE			
`NA2_BKDIAG`		x	x		O	Security exit diagnostics. Use only if requested by Software AG support.	
`NA2_BKPRIV`		x	x		O	Security exit setting.	See *Setting up EntireX Security for Broker Kernel* in the UNIX and Windows post-installation documentation.
`REGFILE`			x		R	RGS repository for Software AG Base Technology components under UNIX.	

Using Environment Variables under z/OS

Under CICS, Batch and IMS, use the `SAGTOKEN` Utility to set and delete environment variables. See *`SAGTOKEN` Utility*.

In Com-plete, use the `EXAENV` environment store to set and delete environment variables. See EXAENV Environment Store.

Using Environment Variables under UNIX

The following table shows how to use environment variables with the C, Bourne and Korn shells. For other shells, see your UNIX documentation.

C Shell

Action	Syntax	Example
Set environment variable	`setenv` *`variable value`*	`setenv ERX_TRACELEVEL ADVANCED`
Delete environment variable	`unsetenv` *`variable`*	`unsetenv ERX_TRACELEVEL`

Bourne and Korn Shells

Action	Syntax	Example

Set environment variable	*`variable`* `=` *`value`*
	`export` *`variable`*
		`ERX_TRACELEVEL=ADVANCED`
		`export ERX_TRACELEVEL`

Delete environment variable	`unset` *`variable`*	`unset ERX_TRACELEVEL`

Using Environment Variables under Windows

The following table shows how to use environment variables under Windows:

Action	Syntax	Examples

Set environment variable	`SET` *`variable`* `=` *`value`*	`SET ERX_TRACELEVEL=ADVANCED`
		`SET ETB_STUBLOG=NONE`

Delete environment variable	`SET` *`variable`* `=`	`SET ERX_TRACELEVEL=`

Using Environment Variables under BS2000 (Batch, Dialog)

Environment variables are emulated with SDF variables or, failing that, with job variables.

Replace all underscores in the variable names by hyphens. For example, variable `ETB_STUBLOG` is called `ETB-STUBLOG` under BS2000.

The following table shows how to use job variables under BS2000:

Action	Syntax	Example
Set environment variable	`/CATJV` *`variable`*	`/CATJV ETB-STUBLOG`
	`/SETJV` *`variable`*`,C'`*`value`*`'`	`/SETJV ETB-STUBLOG,C'1'`
Delete environment variable	`/ERAJV` *`variable`*	`/ERAJV ETB-STUBLOG`

Using Environment Variables under z/VSE

Action	Syntax	Examples
Set environment variable	`//SETPARM variable = value`	`//SETPARM STUBLOG=2`
Delete environment variable	Remove `SETPARM` statement	`/* /SETPARM STUBLOG=2`

3 Directories as Used in EntireX

Application Data Directory

Windows

Under Windows, the application data directory is the folder that serves as a common repository for application-specific data.

Example: *C:\Documents and Settings\username\Application Data*

Broker Directory

UNIX

This directory is a subdirectory of the EntireX main directory */<Install_Dir>/EntireX/config/etb/<brokerid>*.

Example: */<Install_Dir>/EntireX/config/etb/ETB001*

Windows

This directory is a subfolder of the EntireX *config* directory *<drive>:\SoftwareAG\EntireX\config\etb\<brokerid>*.

Example: *<drive>:\SoftwareAG\EntireX\config\etb\ETB001*

Broker User Exit Directory

UNIX

This directory is a subdirectory of the EntireX main directory */<Install_Dir>/EntireX/security_exit*.

Windows

This directory is a subfolder of the EntireX main directory, for example: *C:\SoftwareAG\EntireX\security_exit*.

Local Application Data Directory

Windows

The local application data directory is a folder that serves as a common repository for (non-roaming) application-specific data.

Example: *C:\Documents and Settings\username\Application Data*

Trace Directory

Windows

Traces are written into the *..\My Documents\Software AG\EntireX* folder. The location of the folder *My Documents* can be specified by the user. By default it is a subdirectory of the user's *Profile* folder referenced by the `%USERPROFILE%` environment variable.

Example: *C:\Documents And Settings\username\My Documents\Software AG\EntireX*

User's Home Directory

Windows

This folder is also known as the *My Documents* folder. The location of the folder *My Documents* can be specified by the user. By default it is a subdirectory of the *Profile* folder referenced by the `%USERPROFILE%` environment variable.

Example: *C:\Documents And Settings\*`username`*\My Documents*

Working Directory

Windows

This is the directory your application is running in.

Example: *C:\Temp*

EntireX Directory etc

UNIX

This directory is a subdirectory of the EntireX main directory */<Install_Dir>/EntireX/etc*.

Windows

This directory is a subfolder of the EntireX main directory *<drive>:\SoftwareAG\EntireX\etc*.

Example: *C:\<drive>:\SoftwareAG\EntireX\etc*

4 Broker Resource Allocation

The EntireX Broker is a multithreaded application and communicates among multiple tasks in memory pools. If you do not need to restrict the memory expansion of EntireX Broker, we strongly recommend you enable the dynamic memory management in order to handle changing workload appropriately. See ***Dynamic Memory Management*** below. If dynamic memory management is disabled, non-expandable memory is allocated during startup to store all internal control blocks and the contents of messages.

General Considerations

Resource considerations apply to both the global and service-specific levels:

- Dynamic assignment of global resources to services that need them prevents the return of a “Resource Shortage” code to an application when resources are available globally. It also enables the EntireX Broker to run with fewer total resources, although it does not guarantee the availability of a specific set of resources for a particular service.
- Flow control ensures that individual services do not influence the behavior of other services by accident, error, or simply overload. This means that you can restrict the resource consumption of particular services in order to shield the other services.

In order to satisfy both global and service-specific requirements, the EntireX Broker allows you to allocate resources for each individual service or define global resources which are then allocated dynamically to any service that needs them.

The resources in question are the number of conversations, number of servers, plus units of work and the message storage, separated in a long buffer of 4096 bytes and short buffer of 256 bytes. These resources are typically the bottleneck in a system, especially when you consider that non-conversational communication is treated as the special case of “conversations with a single message only” within the EntireX Broker.

Global resources are defined by the parameters in the Broker section of the attribute file. The number of conversations allocated to each service is defined in the service-specific section of the attribute file. Because the conversations are shared by all servers that provide the service, a larger number of conversations should be allocated to services that are provided by more than one server. The number of conversations required is also affected by the number of clients accessing the service in parallel.

Specifying Global Resources

You can specify a set of global resources with no restrictions on which service allocates the resources:

- Specify the global attributes with the desired values.
- Do not specify any additional restrictions. That is, do not provide values for the following Broker-specific attributes:

  ```
  LONG-BUFFER-DEFAULT
  SHORT-BUFFER-DEFAULT
  CONV-DEFAULT
  SERVER-DEFAULT
  ```

- Also, do not provide values for the following server-specific attributes:

  ```
  LONG-BUFFER-LIMIT
  SERVER-LIMIT
  SHORT-BUFFER-LIMIT
  CONV-LIMIT
  ```

Example

The following example defines global resources. If no additional definitions are specified, resources are allocated and assigned to any server that needs them.

```
NUM-CONVERSATION=1000
NUM-LONG-BUFFER=200
NUM-SHORT-BUFFER=2000
NUM-SERVER=100
```

Restricting the Resources of Particular Services

You can restrict resource allocation for particular services in advance:

- Use `CONV-LIMIT` to limit the resource consumption for a specific service.
- Use `CONV-DEFAULT` to provide a default limit for services for which `CONV-LIMIT` is not defined.

Example

In the following example, attributes are used to restrict resource allocation:

```
DEFAULTS=BROKER
NUM-CONVERSATION=1000
CONV-DEFAULT=200

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, CONV-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B, CONV-LIMIT=UNLIM
CLASS=C, SERVER=C, SERVICE=C
```

- Memory for a total of 1000 conversations is allocated (`NUM-CONVERSATION=1000`).
- Service A (`CLASS A`,`SERVER A`,`SERVICE A`) is limited to 100 conversation control blocks used simultaneously (`CONV-LIMIT=100`). The application that wants to start more conversations than specified by the limit policy will receive a "Resource shortage" return code. This return code should result in a retry of the desired operation a little later, when the resource situation may have changed.
- Service B (`CLASS B`,`SERVER B`,`SERVICE B`) is allowed to try to allocate as many resources as necessary, provided the resources are available and not occupied by other services. The number of conversations that may be used by this service is unlimited (`CONV-LIMIT=UNLIM`).
- Service C (`CLASS C`,`SERVER C`,`SERVICE C`) has no explicit value for the `CONV-LIMIT` attribute. The number of conversation control blocks that it is allowed to use is therefore limited to the default value which is defined by the `CONV-DEFAULT` Broker attribute.

The same scheme applies to the allocation of message buffers and servers:

- In the following example, long message buffers are allocated using the keywords `NUM-LONG-BUFFER`, `LONG-BUFFER-DEFAULT` and `LONG-BUFFER-LIMIT`:

  ```
  DEFAULTS=BROKER
  NUM-LONG-BUFFER=2000
  LONG-BUFFER-DEFAULT=250

  DEFAULTS=SERVICE
  CLASS=A, SERVER=A, SERVICE=A, LONG-BUFFER-LIMIT=100
  CLASS=B, SERVER=B, SERVICE=B, LONG-BUFFER-LIMIT=UNLIM
  CLASS=C, SERVER=C, SERVICE=C
  ```

- In the following example, short message buffers are allocated using the keywords `NUM-SHORT-BUFFER`, `SHORT-BUFFER-DEFAULT` and `SHORT-BUFFER-LIMIT`:

  ```
  DEFAULTS=BROKER
  NUM-SHORT-BUFFER=2000
  SHORT-BUFFER-DEFAULT=250

  DEFAULTS=SERVICE
  CLASS=A, SERVER=A, SERVICE=A, SHORT-BUFFER-LIMIT=100
  CLASS=B, SERVER=B, SERVICE=B, SHORT-BUFFER-LIMIT=UNLIM
  CLASS=C, SERVER=C, SERVICE=C
  ```

- In the following example, servers are allocated using the keywords NUM-SERVER, SERVER-DEFAULT and SERVER-LIMIT:

  ```
  DEFAULTS=BROKER
  NUM-SERVER=2000
  SERVER-DEFAULT=250

  DEFAULTS=SERVICE
  CLASS=A, SERVER=A, SERVICE=A, SERVER-LIMIT=100
  CLASS=B, SERVER=B, SERVICE=B, SERVER-LIMIT=UNLIM
  CLASS=C, SERVER=C, SERVICE=C
  ```

Specifying Attributes for Privileged Services

If privileged services (services with access to unlimited resources) exist, specify UNLIMITED for the attributes CONV-LIMIT, SERVER-LIMIT, LONG-BUFFER-LIMIT and SHORT-BUFFER-LIMIT in the service-specific section of the attribute file.

For example:

```
DEFAULTS=SERVICE
CONV-LIMIT=UNLIM
LONG-BUFFER-LIMIT=UNLIM
SHORT-BUFFER-LIMIT=UNLIM
SERVER-LIMIT=UNLIM
```

To ensure a resource reservoir for peak load of privileged services, define more resources than would normally be expected by specifying larger numbers for the Broker attributes that control global resources:

```
NUM-SERVER
NUM-CONVERSATION
CONV-DEFAULT
LONG-BUFFER-DEFAULT
SHORT-BUFFER-DEFAULT
SERVER-DEFAULT
```

Maximum Units of Work

The maximum number of units of work (UOWs) that can be active concurrently is specified in the Broker attribute file. The `MAX-UOWS` attribute can be specified for the Broker globally as well as for individual services. It cannot be calculated automatically. If a service is intended to process UOWs, a `MAX-UOWS` value must be specified.

If message processing only is to be done, specify `MAX-UOWS=0` (zero). The Broker (or the service) will not accept units of work, i.e., it will process only messages that are not part of a UOW. Zero is used as the default value for `MAX-UOWS` in order to prevent the sending of UOWs to services that are not intended to process them.

Calculating Resources Automatically

To ensure that each service runs without impacting other services, allow the EntireX Broker to calculate resource requirements automatically:

- Ensure that the attributes that define the default total for the Broker and the limit for each service are not set to `UNLIM`.
- Specify `AUTO` for the Broker attribute that defines the total number of the resource.
- Specify a suitable value for the Broker attribute that defines the default number of the resource.

The total number required will be calculated from the number defined for each service. The resources that can be calculated this way are Number of Conversations, Number of Servers, Long Message Buffers and Short Message Buffers.

Avoid altering the service-specific definitions at runtime. Doing so could corrupt the conversation consistency. Applications might receive a message such as “`NUM-CONVERSATIONS` reached” although the addressed service does not serve as many conversations as defined. The same applies to the attributes that define the long and short buffer resources.

Automatic resource calculation has the additional advantage of limiting the amount of memory used to run the EntireX Broker. Over time, you should be able to determine which services need more resources by noting the occurrence of the return code “resource shortage, please retry”. You can then increase the resources for these services. To avoid disruption to the user, you could instead allocate a relatively large set of resources initially and then decrease the values using information gained from the Administration Monitor application.

Number of Conversations

To calculate the total number of conversations automatically, ensure that the `CONV-DEFAULT` Broker attribute and the `CONV-LIMIT` service-specific attribute are not set to `UNLIM` anywhere in the attribute file. Specify `NUM-CONVERSATION=AUTO` and an appropriate value for the `CONV-DEFAULT` Broker attribute. The total number of conversations will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-CONVERSATION=AUTO
CONV-DEFAULT=200

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A
CLASS=B, SERVER=B, SERVICE=B, CONV-LIMIT=100
CLASS=C, SERVER=C, SERVICE=C
```

- Service A and Service C both need 200 conversations (the default value). Service B needs 100 conversations (`CONV-LIMIT=100`).
- Because `NUM-CONVERSATIONS` is defined as `AUTO`, the broker calculates a total of 500 conversations (200 + 200 + 100).
- `NUM-CONVERSATIONS=AUTO` allows the number of conversations to be flexible without requiring additional specifications. It also ensures that the broker is started with enough resources to meet all the demands of the individual services.
- `AUTO` and `UNLIM` are mutually exclusive. If `CONV-DEFAULT` or a single `CONV-LIMIT` is defined as `UNLIM`, the EntireX Broker cannot determine the number of conversations to use in the calculation, and the EntireX Broker cannot be started.

Number of Servers

To calculate the number of servers automatically, ensure that the `SERVER-DEFAULT` Broker attribute and the `SERVER-LIMIT` service-specific attribute are not set to `UNLIM` anywhere in the attribute file. Specify `NUM-SERVER=AUTO` and an appropriate value for the `SERVER-DEFAULT` Broker attribute. The total number of server buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-SERVER=AUTO
SERVER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, SERVER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B
CLASS=C, SERVER=C, SERVICE=C
```

Long Message Buffers

To calculate the number of long message buffers automatically, ensure that the `LONG-BUFFER-DEFAULT` Broker attribute and the `LONG-BUFFER-LIMIT` service-specific attribute are not set to `UNLIM` anywhere in the attribute file. Specify NUM-LONG-BUFFER=AUTO and an appropriate value for the LONG-BUFFER-DEFAULT Broker attribute. The total number of long message buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-LONG-BUFFER=AUTO
LONG-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, LONG-BUFFER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B
CLASS=C, SERVER=C, SERVICE=C
```

Short Message Buffers

To calculate the number of short message buffers automatically, ensure that the SHORT-BUFFER-DEFAULT Broker attribute and the SHORT-BUFFER-LIMIT service-specific attribute are not set to UNLIM anywhere in the attribute file. Specify NUM-SHORT-BUFFER=AUTO and an appropriate value for the SHORT-BUFFER-DEFAULT Broker attribute. The total number of short message buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-SHORT-BUFFER=AUTO
SHORT-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A
CLASS=B, SERVER=B, SERVICE=B, SHORT-BUFFER-LIMIT=100
CLASS=C, SERVER=C, SERVICE=C
```

Dynamic Memory Management

Dynamic memory management is a feature to handle changing Broker workload without any restart of the Broker task. It increases the availability of the Broker by using various memory pools for various Broker resources and by being able to use a variable number of pools for the resources.

If more memory is needed than currently available, another memory pool is allocated for the specific type of resource. If a particular memory pool is no longer used, it will be deallocated.

The following Broker attributes can be omitted if DYNAMIC-MEMORY-MANAGEMENT=YES has been defined:

- NUM-CLIENT
- NUM-CMDLOG-FILTER
- NUM-COMBUF
- NUM-CONV[ERSATION]
- NUM-LONG[-BUFFER]
- NUM-SERVER
- NUM-SERVICE
- NUM-SERVICE-EXTENSION
- NUM-SHORT[-BUFFER]
- NUM-UOW|MAX-UOWS|MUOW
- NUM-WQE

If you want statistics on allocation and deallocation operations in Broker, you can configure Broker to create a storage report with the attribute STORAGE-REPORT. See *Storage Report* below.

Note: To ensure a stable environment, some pools of Broker are not deallocated automatically. The first pools of type COMMUNICATION, CONVERSATION, CONNECTION, HEAP, PARTICIPANT, PARTICIPANT EXTENSION, SERVICE ATTRIBUTES, SERVICE, SERVICE EXTENSION, TIMEOUT QUEUE, TRANSLATION, WORK QUEUE are excluded from the automatic deallocation even when they have not been used for quite some time. Large pools cannot be reallocated under some circumstances if the level of fragmentation in the address space has been increased in the meantime.

Dynamic Worker Management

Dynamic worker management is a feature to handle the fluctuating broker workload without restarting the Broker task. It adjusts the number of running worker tasks according to current workload. The initial portion of worker tasks started at Broker startup is still determined by NUM-WORKER.

If more workers are needed than currently available, another worker task is started. If a worker task is no longer needed, it will be stopped.

The following Broker attributes are used for the configuration if DYNAMIC-WORKER-MANAGEMENT=YES has been defined:

- WORKER-MAX
- WORKER-MIN
- WORKER-NONACT
- WORKER-QUEUE-DEPTH
- WORKER-START-DELAY

The following two attributes are very performance-sensitive:

- Attribute WORKER-QUEUE-DEPTH defines the number of unassigned user requests in the input queue before a new worker task is started.

- Attribute `WORKER-START-DELAY` defines the time between the last worker task startup and the next check for another possible worker task startup. It is needed to consider the time for activating a worker task.

Both attributes depend on the environment, in particular the underlying operating system and the hardware. The goal is to achieve high-performance user request processing without starting too many worker tasks.

A good starting point to achieve high performance is not to change the attributes and to observe the performance of the application programs after activating the dynamic worker management.

If broker attribute `DYNAMIC-WORKER-MANAGEMENT=YES` is set, operator commands are available under z/OS to deactivate and subsequently reactivate dynamic worker management.

The following section illustrates the two different modes of dynamic worker management:

- **Scenario 1**

  ```
  DYNAMIC-WORKER-MANAGEMENT=YES
  NUM-WORKER = 5
  WORKER-MIN = 1
  WORKER-MAX = 32
  ```

  Broker is started with 5 worker tasks and then dynamically varies the number of worker tasks within the range from `WORKER-MIN=1` to `WORKER-MAX=32` due to `DYNAMIC-WORKER-MANAGEMENT=YES`.

- **Scenario 2**

  ```
  DYNAMIC-WORKER-MANAGEMENT=NO
  NUM-WORKER = 5
  WORKER-MIN = 1
  WORKER-MAX = 32
  ```

  Broker is started with 5 worker tasks. The `WORKER-MIN/MAX` attributes are ignored due to `DYNAMIC-WORKER-MANAGEMENT=NO`.

Storage Report

You can create an optional report file that provides details about all activities to allocate or to deallocate memory pools. This section details how to create the report and provides a sample report.

- Creating a Storage Report
- Platform-specific Rules
- Sample Storage Report

See also Broker-specific attribute `STORAGE-REPORT`.

Creating a Storage Report

Use Broker's global attribute `STORAGE-REPORT` with the value `YES`. If attribute value `YES` is supplied, all memory pool operations will be reported if the output mechanism is available. If the value `NO` is specified, no report will be created.

Platform-specific Rules

z/OS

`DDNAME ETBSREP` assigns the report file. Format `RECFM=FB, LRECL=121` is used.

UNIX and Windows

Broker creates a file with the name *STORAGE.REPORT* in the current working directory. If the environment variable `ETB_STORAGE_REPORT` is supplied, the file name specified in the environment variable will be used. If Broker receives the command-line argument `-r`, the token following argument `-r` will be used as the file name.

BS2000

`LINK-NAME ETBSREP` assigns the report file. Format `REC-FORM=V, REC-SIZE=0, FILE-TYPE ISAM` is used by default.

z/VSE

Logical unit `SYS015` and logical file name *ETBSREP* are used. Format `RECORD-FORMAT=FB, RECORD-LENGTH=121` is used.

Sample Storage Report

The following is an excerpt from a sample STORAGE report.

```
EntireX 8.1.0.00     STORAGE Report     2009-06-26 12:28:58    Page     1

Identifier     Address        Size         Total          Date      Time    Action
KERNEL POOL  0x25E48010   407184 bytes   407184 bytes  2009-06-26 12:...  Allocated
HEAP POOL    0x25EB4010  1050692 bytes  1457876 bytes  2009-06-26 12:...  Allocated
...
```

Header	Description
Identifier	Name of the memory pool.
Address	Start address of the memory pool.
Size	Size of the memory pool.
Total	Total size of all obtained memory pools.
Date, Time	Date and time of the action.
Action	The action of Broker. The following actions are currently supported: Allocated: memory pool is allocated. Deallocated: memory pool is deallocated.

Maximum TCP/IP Connections per Communicator

This table shows the generated maximum number of TCP/IP connections per communicator. See also:

- Note for z/OS
- Note for Linux

Platform	Maximum Number of TCP/IP Connections per Communicator
AIX	2,048
BS2000	2,048
Linux	65,534
Solaris	65,356
Windows	4,096
z/OS	16,384
z/VSE	2,048

With the Broker-specific attribute `POLL`, these restrictions can be lifted under z/OS, UNIX and z/VSE. See `POLL`.

The number of communicators multiplied by the maximum number of connections cannot exceed the maximum number of file descriptors per process.

See also `MAX-CONNECTIONS` under `TCP-OBJECT (Struct INFO_TCP)` under *Broker CIS Data Structures* in the EntireX Broker ACI Programming documentation.

Note for z/OS

Under z/OS, the following message may appear in the broker log:

```
ETBD0286 Diagnostic Values:
 accept: 124, EDC5124I Too many open files.errno2: 84607302 050B0146
```

The most common reason for this TCP/IP Communicator diagnostic message is the limitation of open files per user. The value of `MAXFILEPROC` in the `BPXPRM00` parmlib member should be greater than the expected number of TCP/IP connections.

Note for Linux

Under Linux, setting the maximum open file limit depends on your working environment:

- bash
- systemd

bash

In the bash shell you can display or change the limits with the command `ulimit -n`. These limits are used when the Broker (etbnuc) is started from the command line or from a cron job.

The limits can be stored, for example, in the file */etc/security/limits.conf*.

- For all users:

```
* soft nofile 1024
* hard nofile 8192
```

- For user entirex:

```
entirex soft nofile   8192
entirex hard nofile 100000
```

Broker uses the soft limit. When this limit is reached, no more connections are possible. If the hard limit is higher than the soft limit, you can increase the limit - without having to stop the broker - using the following command:

```
#> prlimit --pid <pid> --nofile = 4096:8192
```

The maximum limit in the broker for `POLL=NO` is 65534. `POLL=YES` is not subject to any limit and is dependent only on the soft limit of the system.

systemd

If the broker is controlled by a service that was started by `systemd`, the limits of `systemd` apply.

There are various ways of increasing the limits if you need more than 4096 connections:

- Set `DefaultLimitNOFILE` in the files */etc/systemd/system.conf* or */etc/systemd/user.conf*.
- Insert `LimitNOFILE=<new-limit>` in a service file */usr/lib/systemd/system/sag<n>exx<vers>*.
  Example:

```
# -----------------------------------------------------------------------------
# Copyright (c) 2014-2020 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.
# Use, reproduction, transfer, publication or disclosure is prohibited except
# as specifically provided for in your License Agreement with Software AG.
# -----------------------------------------------------------------------------
# do not modify this line
[Unit]
Description=sag7exx105
After=multi-user.target

[Service]
Type=forking
RemainAfterExit=yes
PrivateTmp=no
KillMode=none
TimeoutStartSec=330
TimeoutStopSec=330
LimitNOFILE=32000
User=rdsadmin
Group=rdstst
ExecStart=/bin/sh -c "/opt/testenv/exx/105/installed/EntireX/bin/sagexx105 start"
ExecStop=/bin/sh -c "/opt/testenv/exx/105/installed/EntireX/bin/sagexx105 stop"
PIDFile=/opt/testenv/exx/v105/installed/EntireX/bin/sagexx105.pid

[Install]
WantedBy=multi-user.target
```

You can check the current settings using the proc file system:

```
#> cat /proc/<etbnuc-pid>/limits

Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             15709                15709                processes
Max open files            32000                32000                files
Max locked memory         65536                65536                bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       15709                15709                signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

5 Broker Attributes

Note: This section lists all EntireX Broker parameters. Not all parameters are applicable to all supported operating systems.

The Broker attribute file contains a series of parameters (attributes) that control the availability and characteristics of clients and servers, as well as of the Broker itself. You can customize the Broker environment by modifying the attribute settings.

Name and Location of Attribute File

The name and location of the broker attribute file is platform-dependent.

Platform	File Name/Location
z/OS	Member *EXBATTR* in the EntireX Broker source library.
UNIX	File *etbfile* in directory <*InstDir*>/EntireX/config/etb/<*BrokerName*> (default) *
Windows	File <*BrokerName*>.atr in directory <*InstDir*>\EntireX\config\etb\<*BrokerName*> (default) *
BS2000	File *ETB-ATTR* in library *EXX103.JOBS*.
z/VSE	Library member *ETBnnn.ATR*, where *nnn* is a placeholder specifying the broker instance (e.g. *nnn*= the assigned broker ID).

* When starting a broker manually, name and location of the broker attribute file can be overwritten with the environment variable `ETB_ATTR`.

Attribute Syntax

Each entry in the attribute file has the format:

```
ATTRIBUTE-NAME=value
```

The following rules and restrictions apply:

- A line can contain multiple entries separated by commas.
- Attribute names can be entered in mixed upper and lowercase.
- Spaces between attribute names, values and separators are ignored.
- Spaces in the attribute names are not allowed.
- Commas and equal signs are not allowed in value notations.
- Lines starting with an asterisk (*) are treated as comment lines. Within a line, characters following an * or # sign are also treated as comments.
- The `CLASS` keyword must be the first keyword in a service definition.
- Multiple services can be included in a single service definition section. The attribute settings will apply to all services defined in the section.
- Attributes specified after the service definition (`CLASS`, `SERVER`, `SERVICE` *keywords*) overwrite the default characteristics for the service.
- Attribute values can contain variables of the form `${`*variable name*`}` or `$`*variable name*:

- Due to variations in EBCDIC codepages, braces should only be used on ASCII (UNIX or Windows) platforms or EBCDIC platforms using the IBM-1047 (US) codepage.
- The variable name can contain only alphanumeric characters and the underscore (_) character.
- The first non-alphanumeric or underscore character terminates the variable name.
- Under UNIX and Windows, the string `${`*`variable name`*`}` is replaced with the value of the corresponding environment variable.
- On z/OS, variable values are read from a file defined by the DD name `ETBVARS`. The syntax of this file is the same as the attribute file.
- If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$`*`variable name`* will be used as the variable value.
- If you encounter problems with braces (and this is quite possible in a z/OS environment), we suggest you omit the braces.

Broker-specific Attributes

The broker-specific attribute section begins with the keyword `DEFAULTS=BROKER`. It contains attributes that apply to the broker. At startup time, the attributes are read and duplicate or missing values are treated as errors. When an error occurs, the broker stops execution until the problem is corrected.

Tip: To avoid resource shortages for your applications, be sure to specify sufficiently large values for the broker attributes that define the global resources.

		Opt/	Operating System				
Attribute	**Values**	**Req**	**z/OS**	**UNIX**	**Windows**	**z/VSE**	**BS2000**
`ABEND-LOOP-DETECTION`	`YES` | `NO`	O	z	u	w	v	b
	`YES` Stop broker if a task terminates abnormally twice, that is, the same abend reason at the same abend location already occurred. This attribute prevents an infinite abend loop. `NO` Use only if requested by Software AG Support. This setting may make sense if a known error leads to an abnormal termination, but a hotfix solving the problem has not yet been provided. Reset to `YES` when the hotfix has been installed.						
`ABEND-MEMORY-DUMP`	`YES` | `NO`	O	z	u	w	v	b
	`YES` Print all data pools of the broker if a task terminates abnormally. This dump is needed to analyze the abend. `NO` If the dump has already been sent to Software AG, you can set to `NO` to avoid the extra overhead.						
`ACCOUNTING`	`NO` | `128-255`	O	z				
	`NO`|`YES[SEPARATOR=`*`char`*`]`	O		u	w	v	b
	Determines whether accounting records are created. `NO` Do not create accounting records. *`nnn`* The SMF record number to use when writing the accounting records. `YES` Create accounting data. *`char`*=separator character(s). Up to seven separator characters can be specified using the `SEPARATOR` suboption, for example: `ACCOUNTING = (YES, SEPARATOR=;)` If no separator character is specified, the comma character will be used. See also *Accounting in EntireX Broker* in the platform-specific Administration documentation.						

		Opt/	Operating System				
Attribute	Values	Req	z/OS	UNIX	Windows	z/VSE	BS2000
ACCOUNTING-VERSION	1 | 2 | 3 | 4 | 5	O	z	u	w	v	b
	Determines whether accounting records are created. 1 Collect accounting information. This value is supported for reasons of compatibility with EntireX Broker 7.2.1 and below. 2 Collect extended accounting information in addition to that available with option 1. 3 Create accounting records in layout of version 3. 4 Create accounting records in layout of version 4. 5 Create accounting records in layout of version 5. This parameter applies when ACCOUNTING is activated.						
ACI-CONVERSION	YES | NO	O	z	u	w	v	b
	Determines the handling of ACI request and response strings of USTATUS. YES Convert ACI request and response strings with ICU. See *ICU Conversion* in the Internationalization documentation. NO Translate ACI request and response with internal translation table without support of national characters. See *Translation User Exit* in the Internationalization documentation. **Note:** This attribute was undocumented in earlier EntireX versions and had default value NO. This meant that a translation user exit was used instead; this is no longer recommended.						
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	v	b
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See *Application Monitoring*.						
AUTOLOGON	YES | NO	O	z	u	w	v	b
	YES LOGON occurs automatically during the first SEND or REGISTER. NO The application has to issue a LOGON call.						
AUTOSTART	NO | YES	O		u	w		
	This attribute defines the autostart behavior of a broker. NO Broker is *not* started automatically with the next system start. YES Broker is restarted automatically with the next system start.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Note: Prior to EntireX version 10.5 this was handled by the Broker Administration Service.						
BLACKLIST-PENALTY-TIME	5m | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Define the length of time a participant is placed on the PARTICIPANT-BLACKLIST to prevent a denial-of-service attack. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific broker Administration documentation.						
BROKER-ID	A32	R	z	u	w	v	b
	Identifies the broker to which the attribute file applies. The broker ID must be unique per machine. **Note:** The numerical section of the BROKER-ID is no longer used to determine the DBID in the EntireX Broker kernel with Entire Net-Work transport (NET). To determine the DBID, use attribute NODE in the DEFAULTS=NET section of the attribute file.						
CLIENT-NONACT	15M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Define the non-activity time for clients. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). A client that does not issue a broker request within the specified time limit is treated as inactive and all resources for the client are freed.						
CMDLOG	NO | YES	O	z	u	w	v	b
	NO Command logging will not be available in the broker. YES Command logging features will be available in the broker.						
CMDLOG-FILE-SIZE	1024 | *n*	O	z	u	w	v	b
	Defines the maximum size of the file that the command log is written to, in kilobytes. The value must be 1024 or higher. The default value is 1024. When one command log file grows to this size, broker starts writing to the other file. For more details, see *Command Logging in EntireX*.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CONTROL-INTERVAL	60s | *n* | *n*S | *n*M | *n*H |	O	z	u	w	v	b
	Defines the time interval of time-driven broker-to-broker calls. 1. It controls the time between handshake attempts. 2. The standby broker will check the status of the standard broker after the elapsed CONTROL-INTERVAL time. *n* Same as *n*S. *n*S Interval in seconds (max. 2147483647). *n*M Interval in minutes (max. 35791394). *n*H Interval in hours (max. 596523). The minimum value is 16 seconds. We strongly recommend the default value (60 seconds), except for very slow machines.						
CONV-DEFAULT	UNLIM | *n*	O	z	u	w	v	b
	Default number of conversations that are allocated for every service. UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM-CONVERSATION. *n* Number of conversations. This value can be overridden by specifying a CONV-LIMIT for the service. A value of 0 (zero) is invalid.						
DEFERRED	NO | YES	O	z	u	w	v	b
	Disable or enable deferred processing of units of work. NO Units of work cannot be sent to the service until it is available. YES Units of work can be sent to a service that is not up and registered. They will be processed when the service becomes available.						
DYNAMIC-MEMORY-MANAGEMENT	YES | NO	O	z	u	w	v	b
	YES An initial portion of memory is allocated at broker startup based on defined NUM-* attributes or internal default values if no NUM-* attributes have been defined. More memory is allocated without broker restart if there is a need to use more storage. Unused memory is deallocated. The upper limit of memory consumption can be defined by the attribute MAX-MEMORY. See *Dynamic Memory Management* under *Broker Resource Allocation*. NO All memory is allocated at broker startup based on the calculation from the defined NUM-* attributes. Size of memory cannot be changed. This was the known behavior of EntireX 7.3 and earlier.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ HEAP-SIZE ■ LONG-BUFFER-DEFAULT ■ SERVER-DEFAULT ■ SHORT-BUFFER-DEFAULT ■ NUM-CLIENT ■ NUM-CMDLOG-FILTER ■ NUM-COMBUF ■ NUM-CONV[ERSATION] ■ NUM-LONG[-BUFFER] ■ NUM-SERVER ■ NUM-SERVICE-EXTENSION ■ NUM-SERVICE ■ NUM-SHORT[-BUFFER] ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-WQE **Caution:** However, if one of these attributes is defined, it determines the allocation size of that particular broker resource.						
DYNAMIC-WORKER-MANAGEMENT	NO | YES	O	z	u	w		b
	NO All worker tasks are started at broker startup. The number of worker tasks is defined by NUM-WORKER. After this initial step, no further worker tasks can be started. This is default and simulates the behavior of EntireX version 8.0 and earlier. YES As above, the initial portion of worker tasks started at broker startup is determined by NUM-WORKER. However, if there is a need to handle an increased workload, additional worker tasks can be started at runtime without restarting broker. Conversely, if a worker task remains unused, it is stopped. The upper and lower limit of running worker tasks can be defined by the attributes WORKER-MIN and WORKER-MAX. If you run broker with DYNAMIC-WORKER-MANAGEMENT=YES, the following attributes are useful to optimize the overall processing: ■ WORKER-MAX ■ WORKER-MIN ■ WORKER-NONACT ■ WORKER-QUEUE-DEPTH ■ WORKER-START-DELAY The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See *Dynamic Worker Management*.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
ETBCOM	YES | NO	O					b
	Bundles the output of the various broker tasks in task ETBCOM.						
FORCE	NO | YES	O		u			
	NO Go down with error if IPC resources still exist. YES Clean up the left-over IPC resources of a previous run. **Note:** 1. If broker is started twice, the second instance will kill the first by removing the IPC resources. 2. For z/OS, z/VSE and BS2000, see separate attribute FORCE under *DEFAULTS=NET*.						
HEAP-SIZE	1024 | *n*	O	z	u	w	v	b
	Defines the size of the internal heap in KB. Not required if you are using DYNAMIC-MEMORY-MANAGEMENT. If you are *not* using dynamic memory management, we strongly recommend specifying - as a minimum - the default value of 1024 KB.						
ICU-CONVERSION	YES | NO	O	z	u	w	v	b
	Disable or enable ICU conversion. Default for z/VSE is NO; other platforms YES. YES ICU is loaded and available for conversion. It is a prerequisite for CONVERSION=SAGTCHA and CONVERSION=SAGTRPC. NO ICU is not loaded and not available for conversion. CONVERSION=SAGTCHA and CONVERSION=SAGTRPC cannot be used. If any of the broker service definitions uses the character conversion approach *ICU Conversion*, that is, CONVERSION=SAGTCHA or CONVERSION=SAGTRPC, ICU-CONVERSION must be set to YES. If you are using only a user exit (see *User Exits*) or CONVERSION=NO as character conversion approach for all your broker service definitions, ICU-CONVERSION can be set to NO. ICU requires additional storage to run properly. If ICU conversion is not needed, setting ICU-CONVERSION to NO will help to avoid unnecessary storage consumption.						
ICU-DATA-DIRECTORY	Folder or directory name in quotes.	O	z	u	w		
	The location where the broker searches for ICU custom converters. See *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation.						
ICU-SET-DATA-DIRECTORY	YES | NO	O	z	u	w		
	Disable or enable ICU custom converter usage.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	YES The broker tries to locate ICU custom converters with the mechanism defined by the platform, see *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. NO Use of ICU custom converters is not possible.						
IPV6	YES | NO	O	z	u	w		b
	YES Establish SSL and TCP/IP transport in IPv6 and IPv4 networks according to the TCP/IP stack configuration. NO Establish SSL and TCP/IP transport in IPv4 network only. This attribute applies to EntireX version 9.0 and above.						
LONG-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	v	b
	Number of long buffers to be allocated for each service. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-LONG-BUFFER. *n* Number of buffers. This value can be overridden by specifying a LONG-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.						
MAX-MEMORY	0 | *n* | *n*K | *n*M | *n*G | UNLIM	O	z	u	w	v	b
	Defines the upper limit of memory allocated by broker if DYNAMIC-MEMORY-MANAGEMENT=YES has been defined. 0, UNLIM No memory limit. others Defines the maximum limit of allocated memory. If limit is exceeded, error 671 “Requested allocation exceeds MAX-MEMORY” is generated.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	v	b
	Maximum message size that the broker kernel can process. This value is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.						
MAX-MESSAGES-IN-UOW	16 | *n*	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
MAX-TRACE-FILES	4 | *n*	O		u	w		
	Defines the number of backup copies of the trace file ETB.LOG. Minimum number is 1; maximum is 999. A new trace file is allocated when the value for TRACE-FILE-SIZE is exceeded. These two attributes prevent a constantly growing						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	ETB.LOG file. See *Trace File Handling* in the UNIX and Windows Administration documentation.						
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.						
MAX-UOWS	0 | *n*	O	z	u	w	v	b
	The maximum number of UOWs that can be concurrently active broker-wide. The default value is 0 (zero), which means that the broker will process only messages that are not part of a unit of work. If UOW processing is to be done by any service, a MAX-UOWS value must be 1 or larger for the broker. The MAX-UOWS value for the service will default to the value set for the broker. NUM-UOW is an alias of this parameter.						
MESSAGE-CASE	NONE | UPPER | LOWER	O	z	u	w	v	b
	Indicates if certain error message texts returned by the broker to its clients or written by the broker to its log file are to be in mixed case, uppercase, or lowercase. NONE No changes are made to message case. UPPER Messages are changed to uppercase. LOWER Messages are changed to lowercase.						
MUOW	See NUM-UOW.						
NEW-UOW-MESSAGES	YES | NO	O	z	u	w	v	b
	YES New UOW messages are allowed. NO New UOW messages are not allowed. This applies to UOW when using Persistence and should not be used for non-persistent UOWs. A usage example could be the following: The broker persistent store reaches capacity and the broker shuts down. You can set NEW-UOW-MESSAGES to NO to prevent new UOW messages from being added after a broker restart. This action allows only consumption (not production) of UOWs to occur after broker restart. After the persistent store capacity has been sufficiently reduced, the EntireX Broker administrator can issue a CIS command, see ALLOW-NEWUOWMSGS. This action allows new UOW messages to be sent to the broker. Reset attribute NEW-UOW-MESSAGES to YES, which permits new UOW messages to be produced in subsequent broker sessions.						
NUM-BLACKLIST-ENTRIES	256 | *n*	O	z	u	w	v	b
	Number of entries in the participant blacklist. Default value is 256 entries. Together with BLACKLIST-PENALTY-TIME and PARTICIPANT-BLACKLIST, this attribute is used to protect a broker running with SECURITY=YES against denial-of-service attacks. See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific broker Administration documentation.						
NUM-CLIENT	*n*	R	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Number of clients that can access the broker concurrently. A value of 0 (zero) is invalid.						
NUM-CMDLOG-FILTER	1 | *n*	O	z	u	w	v	b
	Maximum number of filters that can be specified simultaneously. **Tip:** We recommend you limit this value to the number of services that are being monitored. Minimum value is 1. A value of zero is invalid when the attribute CMDLOG is set to YES. See *Command Logging in EntireX* for more information.						
NUM-COMBUF	1024 | 1-999999	R	z	u	w	v	b
	Determines the maximum number of communication buffers available for processing commands arriving in the broker kernel. The size of one communication buffer is usually 16 KB split into 32 slots of 512 bytes, but it ultimately depends on the hardware architecture of your CPU. A value of 0 (zero) is invalid.						
NUM-CONVERSATION or NUM-CONV	*n* | AUTO	R	z	u	w	v	b
	Defines the number of conversations that can be active concurrently. The number specified should be high enough to account for both conversational and non-conversational requests. (Non-conversational requests are treated internally as one-conversation requests.) *n* Number of conversations. AUTO Uses the CONV-DEFAULT and the service-specific CONV-LIMIT values to calculate the number of conversations. The values used in the calculation must not be set to UNLIM. **Note:** 1. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See *Wildcard Service Definitions*.						
NUM-LONG-BUFFER or NUM-LONG	4096 | *n* | AUTO	R	z	u	w	v	b
	Defines the number of long message containers. Long message containers have a fixed length of 4096 bytes and are used to store requests that are larger than 2048 bytes. Storing a request of 8192 bytes, for example, would require two long message containers. *n* Number of buffers. AUTO Uses the LONG-BUFFER-DEFAULT and the service-specific LONG-BUFFER-LIMIT values to calculate the number of long message buffers. The values used in the calculation must not be set to UNLIM. A value of 0 (zero) is invalid.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request. In *conversational* mode, the last message received is always kept until a new one is received. **Note:** 1. If a catch-all service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See ***Wildcard Service Definitions***.						
NUM-PARTICIPANT-EXTENSION	*n*	O	z	u	w	v	b
	Defines the number of participant extensions to link participants as clients and servers. *n* Number of participant extensions. *not specified* If this attribute is not set, the default value is calculated based on NUM-CLIENT and NUM-SERVER. A value of 0 (zero) is invalid.						
NUM-SERVER	*n* | AUTO	R	z	u	w	v	b
	Defines the number of servers that can offer services concurrently using the broker. This is *not* the number of services that can be registered to the broker (see NUM-SERVICE). *n* Number of servers. AUTO Uses the SERVER-DEFAULT and the service-specific SERVER-LIMIT values to calculate the number of servers. The values used in the calculation must not be set to UNLIM. **Note:** 1. Setting this value higher than the number of services allows the starting of server replicas that provide the same service. 2. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 3. See ***Wildcard Service Definitions***.						
NUM-SERVICE	*n*	R	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Defines the number of services that can be registered to the broker. This is *not* the number of servers that can offer the services (see `NUM-SERVER`). A value of 0 (zero) is invalid.						
`NUM-SERVICE-EXTENSION`	*n* | `AUTO`	O	z	u	w	v	b
	Defines the number of service extensions to link servers to services. *n* Number of service extensions. `AUTO` Uses the value specified or calculated for `NUM-SERVER` + `NUM-CLIENT`, plus an extra cushion. *not specified* If this attribute is not set, the default value is `NUM-SERVER` multiplied by `NUM-SERVICE`. The minimum value is `NUM-SERVER`. The maximum value is `NUM-SERVER` multiplied by `NUM-SERVICE`. Caution is recommended with this attribute: ■ Set this attribute only if the storage resources allocated for service extensions need to be restricted. ■ Note that the value *n* allows only the specified number of server instances of *n* to be used. ■ Value `AUTO` will calculate the number of allowed server instances from `NUM-SERVER`, which itself might be set to `AUTO`. In this case, this also considers the value of `SERVER-DEFAULT` and even the individual `SERVER-LIMIT` for each service definition.						
`NUM-SHORT-BUFFER` or `NUM-SHORT`	*n* | `AUTO`	R	z	u	w	v	b
	Defines the number of short message containers. Short message containers have a fixed length of 256 bytes and are used to store requests of no more than 2048 bytes. To store a request of 1024 bytes, for example, would require four short message containers. *n* Number of buffers. `AUTO` Uses the `SHORT-BUFFER-DEFAULT` and the service-specific `SHORT-BUFFER-LIMIT` values to calculate the number of short message buffers. The values used in the calculation must not be set to `UNLIM`. **Note:** 1. In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request.						

<table>
<tr><th rowspan="2">Attribute</th><th rowspan="2">Values</th><th rowspan="2">Opt/
Req</th><th colspan="5">Operating System</th></tr>
<tr><th>z/OS</th><th>UNIX</th><th>Windows</th><th>z/VSE</th><th>BS2000</th></tr>
<tr><td></td><td colspan="7">2. In <i>conversational</i> mode, the last message received is always kept until a new one is received.
3. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid.
4. See <i>Wildcard Service Definitions</i>.</td></tr>
<tr><td rowspan="2">NUM-UOW</td><td>0 | <i>n</i></td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">The maximum number of UOWs that can be concurrently active broker-wide. The default value is 0 (zero), which means that the broker will process only messages that are not part of a unit of work. If UOW processing is to be done by any service, a NUM-UOW value must be 1 or larger for the broker. (MAX-UOWS is an alias for this attribute.)
The NUM-UOW value for the service will default to the value set for the broker.</td></tr>
<tr><td rowspan="2">NUM-WORKER</td><td>1 | <i>n</i> (max. 10)</td><td>R</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Number of worker tasks that the broker can use. The number of worker tasks determines the number of functions (SEND, RECEIVE, REGISTER, etc.) that can be processed concurrently. At least one worker task is required; this is the default value.</td></tr>
<tr><td rowspan="2">NUM-WQE</td><td>1-32768</td><td>R</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Maximum number of requests that can be processed by the broker in parallel, over all transport mechanisms.
Each broker command is assigned a worker queue element, regardless of the transport mechanism being used. This element is released when the user has received the results of the command, including the case where the command has timed out.</td></tr>
<tr><td rowspan="2">PARTICIPANT-BLACKLIST</td><td>YES | NO</td><td>R</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Determines whether participants attempting a denial-of-service attack on the broker are to be put on a blacklist.
YES Create a participant blacklist.
NO Do not create a participant blacklist.
See <i>Protecting a Broker against Denial-of-Service Attacks</i> in the platform-specific broker Administration documentation.</td></tr>
<tr><td rowspan="2">PARTNER-CLUSTER-ADDRESS</td><td>A32</td><td>R</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">This is the address of the load/unload broker in transport-method-style. Transport methods TCP and SSL are supported. See <i>Transport-method-style Broker ID</i> for more details. This attribute is required if the attribute RUN-MODE is specified.</td></tr>
<tr><td>PERCENTAGE-FOR-CONNECTION-</td><td>90 | 1-100</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
</table>

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SHORTAGE-MESSAGE	Broker will issue a message if the defined percentage value of TCP/IP connections (available file descriptors) is exceeded. Default is 90 percent of the available file descriptors.						
POLL	YES | NO	O	z	u		v	
	In earlier EntireX versions, the maximum number of TCP/IP connections per communicator was limited; see *Maximum TCP/IP Connections per Communicator* for platform-specific list. With attribute POLL introduced in EntireX version 9.0, this restriction can be lifted under z/OS, UNIX and z/VSE. NO This setting is used to run the compatibility mode in Broker. The poll() system call is not used. The limitations described under *Maximum TCP/IP Connections per Communicator* apply. YES The poll() system call is used to lift the resource restrictions with select() in multiplexing file descriptor sets. **Note:** The maximum number of file descriptors per process is a hard limit that cannot be exceeded by POLL=YES. Setting this attribute to YES increases CPU consumption. POLL=YES is only useful if - you need more than the maximum number of TCP/IP connections per communicator, as described under *Maximum TCP/IP Connections per Communicator*, and - this maximum number is less than the maximum number of file descriptors per process We recommend POLL=NO to reduce CPU consumption.						
PSTORE	NO | HOT | COLD	O	z	u	w	v	b
	Defines the status of the persistent store at broker startup, including the condition of persistent units of work (UOWs). With any value other than NO, PSTORE-TYPE must be set. NO No persistent store. HOT Persistent UOWs are restored to their prior state during initialization. COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty. **Note:** For a hot or cold start, the persistent store must be available when your broker is restarted.						
PSTORE-REPORT	NO | YES	O	z	u	w	v	b
	Determines whether PSTORE report is created.						

<table>
<tr><th rowspan="2">Attribute</th><th rowspan="2">Values</th><th rowspan="2">Opt/ Req</th><th colspan="5">Operating System</th></tr>
<tr><th>z/OS</th><th>UNIX</th><th>Windows</th><th>z/VSE</th><th>BS2000</th></tr>
<tr><td></td><td colspan="7">NO Do not create the PSTORE report file.
YES Create the PSTORE report file.
See also *Persistent Store Report*.</td></tr>
<tr><td>PSTORE-TYPE</td><td>DIV (z/OS) |
CTREE (UNIX, Windows) |
ADABAS (all platforms) |
FILE (UNIX, Windows)</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td></td><td colspan="7">Describes the type of persistent store driver required.
DIV Data in Virtual. z/OS only, and default on this platform. See *DIV-specific Attributes* below and *Implementing a DIV Persistent Store* under *Managing the Broker Persistent Store*.
CTREE c-tree database. UNIX and Windows only. See *c-tree-specific Attributes* and *c-tree Database as Persistent Store* in the UNIX and Windows Administration documentation.
ADABAS Adabas. All platforms. See also *Adabas-specific Attributes* (below) and *Managing the Broker Persistent Store* in the platform-specific Administration documentation.
FILE B-Tree database. UNIX and Windows only. No longer supported.</td></tr>
<tr><td>PSTORE-VERSION</td><td>2 | 3 | 4 | 5</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td></td><td colspan="7">Determines the version of the persistent store. PSTORE=COLD is not needed to upgrade the PSTORE to version 3. Any broker restart with PSTORE-VERSION=3 will upgrade the PSTORE version.
PSTORE-VERSION=3 is needed for ICU support.
The DIV PSTORE requires PSTORE-VERSION=4.
PSTORE-VERSION=5 was added in EntireX version 10.1 to support 64-bit time values on z/OS, and unique message IDs on all platforms. See *Unique Message ID*. PSTORE-VERSION=5 significantly improvement Adabas PSTORE performance on all platforms. We strongly recommend you use this version.
Caution:
■ If you go back to PSTORE-VERSION=2 after upgrading to PSTORE-VERSION=3, the broker will only process data previously created with version 2. No version 3 data will be accessible.
■ If you change the DIV PSTORE from version 3 to 4, perform a COLD restart for the change to take effect, or run PSTORE UNLOAD/LOAD first.
■ If you change to PSTORE-VERSION=5, perform a COLD restart for the change to take effect.</td></tr>
</table>

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
RUN-MODE	STANDARD | STANDBY | PSTORE-LOAD | PSTORE-UNLOAD	O	z	u	w	v	b
	Determines the initial run mode of the broker. STANDARD Default value. Normal mode. STANDBY Deprecated. Supported for compatibility reasons. PSTORE-LOAD Broker will run as load broker to write Persistent Store data to a new persistent store. See also *Migrating the Persistent Store*. PSTORE-UNLOAD Broker will run as unload broker to read an existing persistent store and pass the data to a broker running in PSTORE-LOAD mode. See also *Migrating the Persistent Store*.						
SECURITY	NO | YES	O	z	u	w	v	b
	Determines whether EntireX Security is activated. NO EntireX Security is not activated. YES EntireX Security is activated. See *EntireX Security*.						
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	v	b
	Default number of servers that are allowed for every service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO. This value can be overridden by specifying a SERVER-LIMIT for the service. A value of 0 (zero) is invalid.						
SERVICE-UPDATES	YES | NO	O	z	u	w	v	b
	Switch on/off the automatic update mode of the broker. YES The broker reads the attribute file whenever a service registers for the first time. This allows the broker to honor modifications in the attribute file *without* a restart. The attribute file is read only when the first server registers for a particular service; it is not reread when a second replica is activated. NO The attribute file is read only once during broker startup. Any changes to the attribute file will be honored only if the broker is restarted.						
SHORT-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	v	b
	Number of short buffers to be allocated for each service.						

Attribute	Values	Opt/Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-SHORT-BUFFER=AUTO. *n* Number of buffers. This value can be overridden by specifying a SHORT-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.						
STORAGE-REPORT	NO | YES	O	z	u	w	v	b
	Create a storage report about broker memory usage. NO Do not create the storage report. YES Create the storage report. See *Storage Report*.						
STORE	OFF | BROKER	O	z	u	w	v	b
	Sets the default STORE attribute for all units of work. This attribute can be overridden by the STORE field in the Broker ACI control block. OFF Units of work are not persistent. BROKER Units of work are persistent.						
TRACE-DD	A255	O	z				
	A string containing data set attributes enclosed in quotation marks. These attributes describe the trace output file and must be defined if you are using using a GDG (generation data group) as output data set. See *Flushing Trace Data to a GDG Data Set* under *Tracing EntireX Broker*. The following keywords are supported as part of the TRACE-DD value: ■ DATACLAS ■ DCB including BLKSIZE, DSORG, LRECL, RECFM ■ DISP ■ DSN ■ MGMTCLAS ■ SPACE ■ STORCLAS ■ UNIT Refer to your JCL Reference Manual for a complete description of the syntax. Example:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

	`TRACE-DD = "DSNAME=EXX.GDG,`
	`DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB),`
	`DISP=(NEW,CATLG,CATLG),`
	`SPACE=(CYL,(100,10)),`
	`STORCLAS=SMS"`
	
	Note: If you specify TRACE-DD, you must also specify TRMODE=WRAP and a value for TRBUFNUM for the setting to take effect.						

TRACE-FILE-SIZE	*n* | *n*K | *n*M | *n*G	O		u	w		
	Defines the size of one trace file in kilobytes, megabytes or gigabytes. If this size is exceeded, a new trace file is allocated until the maximum number of trace files specified with MAX-TRACE-FILES is reached. There is no default value. These two parameters help prevent a constantly growing ETB.LOG file. See *Trace File Handling* in the UNIX and Windows Administration documentation.						
TRACE-LEVEL	<u>0</u>-4	O	z	u	w	v	b
	The level of tracing to be performed while the broker is running. 0 No tracing. Default value. 1 Traces incoming requests, outgoing replies, resource usage and conversion errors. 2 All of trace level 1, plus all main routines executed. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus Broker ACI control block displays. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use Command Central or the EntireX Broker command-line utility ETBCMD.						
TRANSPORT	<u>TCP-NET</u> | TCP | SSL | NET	O	z			v	b
	<u>TCP</u> | SSL	O		u	w		
	The broker transport may be specified as any combination of one or more of the following methods: TCP TCP/IP is supported. SSL SSL/TLS is supported. NET Entire Net-Work is supported. This value is not supported for a broker under UNIX or Windows. **Examples:** TRANSPORT=NET specifies that only the Entire Net-Work transport method will be supported by the broker.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	TRANSPORT=TCP-NET specifies that both the TCP/IP and Net-Work transport methods will be supported by the broker. TRANSPORT=TCP-SSL-NET specifies that the TCP/IP, SSL/TLS, and Entire Net-Work transport methods will be supported by the broker. The parameters for each transport method are described in the respective section: **TCP** | **SSL** | **NET**.						
TRAP-ERROR	*nnnn*	O	z	u	w		b
	Where *nnnn* is the four-digit API error number that triggers the trace handler, for example 0007 (Service not registered). Leading zeros are not required. There is no default value. See *Deferred Tracing* in the platform-specific Broker Administration documentation.						
TRBUFNUM	*n*	O	z	u	w		b
	Changes the trace to write trace data to internal trace buffers. *n* is the size of the trace buffer in 64 KB units. There is no default value.						
TRMODE	WRAP	O	z	u	w		b
	Changes the trace mode. WRAP is the only possible value. This value instructs broker to write the trace buffer (see TRBUFNUM) if an event occurs. This event is triggered by a matching TRAP-ERROR during request processing or when an exception occurs.						
UMSG	See MAX-MESSAGES-IN-UOW.						
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	v	b
	Defines the default lifetime for units of work for the service. *n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). If the UOW is inactive - that is, is not processed within the time limit - it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block. See *Timeout Considerations for EntireX Broker*.						
UOW-MSGS	See MAX-MESSAGES-IN-UOW.						
UOW-STATUS-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	v	b
	The value to be added to the UOW-DATA-LIFETIME (lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	*n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523). *n*D Number of days (max. 24855). This attribute is ignored if PSTORE=NO is defined. The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.						
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.						
UWSTATP	0 | *n*	O	z	u	w	v	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-DATA-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1-254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.						
UWTIME	Alias for UOW-DATA-LIFETIME.						
WAIT-FOR-ACTIVE-PSTORE	NO | YES	O	z	u	w	v	b
	Determines whether broker should wait for the Adabas Persistent Store to become active, or until c-tree PSTORE files become available. NO If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will stop. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will stop. YES If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will retry every 10 seconds to initiate						

			Operating System				
Attribute	Values	Opt/ Req	z/OS	UNIX	Windows	z/VSE	BS2000
	communications with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will retry every 10 seconds to rebuild the persistent data. Broker will reject any user requests until it is able to rebuild the persistent data.						
WORKER-MAX	32 | *n* (min. 1, max. 32)	O	z	u	w		b
	Maximum number of worker tasks the broker can use.						
WORKER-MIN	1 | *n* (min. 1, max. 32)	O	z	u	w		b
	Minimum number of worker tasks the broker can use.						
WORKER-NONACT	70S | *n* | *n*S | *n*M | *n*H	O	z	u	w		b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*S. *n*S Non-activity time in seconds (default 70, max. 2147483647). *n*M Non-activity time in in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). **Caution:** A value of 0 (zero) is invalid. If you set this value too low, additional overhead is required for starting and stopping worker tasks. The default and recommended value is 70S.						
WORKER-QUEUE-DEPTH	1 | *n* (min. 1)	O	z	u	w		b
	Number of unassigned user requests in the input queue before another worker task gets started. The default and recommended value is 1. A higher value will result in longer broker response times.						
WORKER-START-DELAY	*internal-value* | *n*	O	z	u	w		b
	n Delay is extended by *n* seconds. Delay after a successful worker task invocation before another worker task can be started to handle current incoming workload. This attribute is used to avoid the risk of recursive invocation of worker tasks, because starting a worker task itself causes workload increase. If no value is specified, an internal value calculated by the broker is used to optimize dynamic worker management. This calculated value is the maximum time required to start a worker task.						

Service-specific Attributes

Each section begins with the keyword DEFAULTS=SERVICE. Services with common attribute values can be grouped together. The attributes defined in the grouping apply to all services specified within it. However, if a different attribute value is defined immediately following the service definition, that new value applies. See also the sections ***Wildcard Service Definitions*** and ***Service Update Modes*** below the table.

Attribute	Values	Opt/Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	v	b
	YES Enable application monitoring for the specified services. NO Disable application monitoring for the specified services. See *Application Monitoring*.						
APPLICATION-MONITORING-NAME or APPMON-NAME	A100	O	z	u	w	v	b
	Specifies the application monitoring name. Used to set the value of the **ApplicationName** KPI. If omitted, the default value from the APPLICATION-MONITORING section is used. If this value is also not specified, the corresponding CLASS/SERVER/SERVICE names are used. See *Application Monitoring*.						
CLASS	A32 (case-sensitive)	R	z	u	w	v	b
	Part of the name that identifies the service together with the SERVER and SERVICE attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. Classes starting with any of the following are reserved for use by Software AG and should not be used in customer-written applications: BROKER, SAG, ENTIRE, ETB, RPC, ADABAS, NATURAL. Valid characters for class name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma. See also the restriction for SERVICE attribute names.						
CLIENT-RPC-AUTHORIZATION	N | Y	O	z				b
	Determines whether this service is subject to RPC authorization checking. N No RPC authorization checking is performed. Y RPC library and program name are appended to the authorization check performed by EntireX Security. Specify YES only to RPC-supported services.						

<table>
<tr><th rowspan="2">Attribute</th><th rowspan="2">Values</th><th rowspan="2">Opt/ Req</th><th colspan="5">Operating System</th></tr>
<tr><th>z/OS</th><th>UNIX</th><th>Windows</th><th>z/VSE</th><th>BS2000</th></tr>
<tr><td></td><td colspan="7">To allow conformity with Natural Security, the CLIENT-RPC-AUTHORIZATION parameter can optionally be defined with a prefix character as follows: CLIENT-RPC-AUTHORIZATION= (YES,<prefix-character&gt;).</td></tr>
<tr><td rowspan="2">CONV-LIMIT</td><td>UNLIM | <i>n</i></td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Allocates a number of conversations especially for this service.
UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM-CONVERSATION=AUTO in the Broker section of the attribute file.
<i>n</i> Number of conversations.
A value of 0 (zero) is invalid.
If NUM-CONVERSATION=AUTO is specified in the Broker section of the attribute file, CONV-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the CONV-LIMIT attribute must be suppressed entirely for the service so that the default (CONV-DEFAULT) becomes active.</td></tr>
<tr><td rowspan="2">CONV-NONACT</td><td>5M | <i>n</i> | <i>n</i>S | <i>n</i>M | <i>n</i>H</td><td>R</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Non-activity time for connections.
<i>n</i> Same as <i>n</i>S.
<i>n</i>S Non-activity time in seconds (max. 2147483647).
<i>n</i>M Non-activity time in minutes (max. 35791394).
<i>n</i>H Non-activity time in hours (max. 596523).
A value of 0 (zero) is invalid. If a connection is not used for the specified time, that is, a server or a client does not issue a broker request that references the connection in any way, the connection is treated as inactive and the allocated resources are freed.</td></tr>
<tr><td rowspan="2">CONVERSION</td><td>A255
(SAGTCHA[,TRACE=<i>n</i>][,<i>OPTION</i>=<i>s</i>] |
SAGTRPC[,TRACE=<i>n</i>][,<i>OPTION</i>=<i>s</i>] |
<i>name</i>[,TRACE=<i>n</i>] |
NO)</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Defines ICU conversion or SAGTRPC user exit for character conversion. See <i>Internationalization with EntireX</i>.
SAGTCHA (1) Conversion using ICU Conversion for <i>ACI-based Programming</i>.
SAGTRPC (2) Conversion using ICU Conversion for <i>RPC-based Components</i> and <i>Reliable RPC</i>.
<i>name</i> (3) Name of the SAGTRPC user exit for RPC-based components and Reliable RPC. See also <i>Configuring SAGTRPC User Exits</i> under</td></tr>
</table>

		Opt/	Operating System				
Attribute	**Values**	**Req**	**z/OS**	**UNIX**	**Windows**	**z/VSE**	**BS2000**

	Configuring Broker for Internationalization in the platform-specific Administration documentation and *Writing SAGTRPC User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation.
	`NO` If conversion is not to be used, either omit the `CONVERSION` attribute or specify `CONVERSION=NO`, for example for binary payload.
	The `CONVERSION` attribute overrides the `TRANSLATION` attribute when defined for a service. That is, when `TRANSLATION` and `CONVERSION` are both defined, `TRANSLATION` will be ignored.
	Note:
	1. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation.
	2. SAGTRPC is not supported on BS2000. For conversion with single-byte code pages, use SAGTCHA on BS2000 for *RPC-based Components* and *Reliable RPC*.
	3. SAGTRPC user exit is not supported on z/VSE and BS2000.
	`TRACE`
	If tracing is switched on, the trace output is written to the broker log file. The following trace levels are available:
	`0` No tracing
	`1 STANDARD` This level is an "on-error" trace. It provides information on conversion errors only. For RPC calls this includes the IDL library, IDL program and the data. Please note that if ***`OPTION` Values for Conversion*** are set, errors are ignored.
	`2 ADVANCED` Tracing of incoming, outgoing parameters and the payload.
	`3 SUPPORT` This trace level is for support diagnostics and should only be switched on when requested by Software AG support.
	`OPTION`
	See table of possible values under ***`OPTION` Values for Conversion***.						

`DEFERRED`	<u>`NO`</u> | `YES`	O	z	u	w	v	b
	`NO` Units of work cannot be sent to the service until it is available. `YES` Units of work can be sent to a service that is not up and registered. The units of work will be processed when the service becomes available.						
`LOAD-BALANCING`	<u>`YES`</u> | `NO`	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	YES When servers that offer a particular service are started, new conversations will be assigned to these servers in a round-robin fashion. The first waiting server will get the first new conversation, the second waiting server will get the second new conversation, and so on. NO A new conversation is always assigned to the first server in the queue.						
LONG-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of long message buffers for the service. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-LONG-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of long message buffers. A value of 0 (zero) is invalid. If NUM-LONG-BUFFER=AUTO is specified in the Broker section of the attribute file, LONG-BUFFER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the LONG-BUFFER-LIMIT attribute must be suppressed entirely for the service so that the default (LONG-BUFFER-DEFAULT) becomes active.						
MAX-MESSAGES-IN-UOW	16 | *n*	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w		b
	Maximum message size that can be sent to a service. This is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.						
MAX-UOWS	0 | *n*	O	z	u	w	v	b
	0 The service does not accept units of work, i.e. it processes only messages that are not part of a UOW. Using zero prevents the sending of UOWs to services that are not intended to process them. *n* Maximum number of UOWs that can be active concurrently for the service. If you do not provide a MAX-UOWS value for the service, it defaults to the MAX-UOWS setting for the broker. If you provide a value that exceeds that of the broker, the service MAX-UOWS is set to the broker's MAX-UOWS value and a warning message is issued. Specify MAX-UOWS=0 for Natural RPC Servers. This restriction will be removed with a later release.						
MUOW	See MAX-UOWS.						

<table>
<tr><th rowspan="2">Attribute</th><th rowspan="2">Values</th><th rowspan="2">Opt/ Req</th><th colspan="5">Operating System</th></tr>
<tr><th>z/OS</th><th>UNIX</th><th>Windows</th><th>z/VSE</th><th>BS2000</th></tr>
<tr><td rowspan="2">NOTIFY-EOC</td><td>NO | YES</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Specifies whether timed-out conversations are to be stored or discarded.
NO Discard the EOC notifications if the server is not ready to receive.
YES Store the EOC notifications if the server is not ready to receive and then notify the server if possible.
If a server is not ready to receive an EOC notification, it can be stored or discarded. If it is stored, the server is notified, if possible, when it is ready to receive.
<b>Caution:</b> The behavior activated by this parameter can be relied upon only during a single lifetime of the broker kernel. Specifically, conversations containing units of work, whose lifetime can span multiple broker kernel sessions, cannot be assumed to show this behavior, even with NOTIFY-EOC=YES.</td></tr>
<tr><td>NUM-UOW</td><td colspan="7">Alias for MAX-UOWS.</td></tr>
<tr><td rowspan="2">POSTPONE-ATTEMPTS</td><td>0 | <i>n</i></td><td>O</td><td>z</td><td>u</td><td>w</td><td></td><td></td></tr>
<tr><td colspan="7">Defines the number of attempts putting a received unit of work (UOW) due to SYNCPOINT option CANCEL on the postpone queue for later processing.
0 All UOWs rejected by the receiver (SYNCPOINT option CANCEL) will be cancelled immediately. Attribute POSTPONE-DELAY is ignored.
<i>n</i> Defines the number of postpone attempts that are performed instead of considering the UOW finished due to SYNCPOINT option CANCEL; the UOW will be moved to the postpone queue and the UOW status will be changed to POSTPONED. These UOWs will be delivered to the receiver when the time specified with POSTPONE-DELAY has elapsed.
The default value is 0. See <i>Postponing Units of Work</i>.</td></tr>
<tr><td rowspan="2">POSTPONE-DELAY</td><td>0 | <i>n</i> | <i>n</i>S | <i>n</i>M | <i>n</i>H</td><td>O</td><td>z</td><td>u</td><td>w</td><td></td><td></td></tr>
<tr><td colspan="7">The length of time a UOW is kept in status POSTPONED.
0 The postpone feature is disabled. Attribute POSTPONE-ATTEMPTS is ignored.
<i>n</i>S Number of seconds the UOW stays unreadable in the postpone queue with status POSTPONED (max. 2147483647).
<i>n</i>M Number of minutes the UOW stays unreadable in the postpone queue with status POSTPONED (max. 35791394).
<i>n</i>H Number of hours the UOW stays unreadable in the postpone queue with status POSTPONED (max. 596523).
<i>n</i>D Number of days the UOW stays unreadable in the postpone queue with status POSTPONED (max. 24855).</td></tr>
</table>

Attribute	Values	Opt/Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	The status of the UOW will be changed from POSTPONED to ACCEPTED after elapsed POSTPONE-DELAY. This delay time does not affect the UOW-DATA-LIFETIME. The POSTPONE-DELAY must be less than UOW-STATUS-LIFETIME in order to make the UOW receivable again. **Note:** By default, the postpone feature is disabled. However, if any value is specified, the minimum delay is 30 seconds. Any value entered that is less than 30 seconds will be increased to this value.						
SERVER	A32 (case-sensitive)	R	z	u	w	v	b
	Part of the name that identifies the service together with the CLASS and SERVICE attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. Valid characters for server name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma.						
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	v	b
	Default number of servers that are allowed for every service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO. A value of 0 (zero) is invalid. This value can be overridden by specifying a SERVER-LIMIT for the service.						
SERVER-LIMIT	*n* | UNLIM	O	z	u	w	v	b
	Allows a number of servers especially for this service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO in the Broker section of the attribute file. A value of 0 (zero) is invalid. If NUM-SERVER=AUTO is specified in the Broker section of the attribute file, SERVER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SERVER-LIMIT attribute must be suppressed entirely for the service so that the default (SERVER-DEFAULT) becomes active. **Note:** UNIX and Windows: This limit also includes any attach server you are using. Make sure you increase the number by one for each attach server you use.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SERVER-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Non-activity time for servers. A server that does not issue a broker request within the specified time limit is treated as inactive and all resources for the server are freed. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). If a server registers multiple services, the highest value of all the services registered is taken as non-activity time for the server.						
SERVICE	A32 (case-sensitive)	R	z	u	w	v	b
	Part of the name that identifies the service together with the CLASS and SERVER attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. The SERVICE attribute names EXTRACTOR and DEPLOYMENT are reserved for Software AG internal use and should not be used in customer-written applications. Valid characters for service name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma. See also the restriction for CLASS attribute names.						
SHORT-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of short message buffers for the service. UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-SHORT-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of short message buffers. If NUM-SHORT-BUFFER=AUTO is specified in the Broker section of the attribute file, SHORT-BUFFER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the service so that the default (SHORT-BUFFER-DEFAULT) becomes active.						
STORE	OFF | BROKER	O	z	u	w	v	b
	Sets the default STORE attribute for all units of work sent to the service. OFF Units of work are not persistent. BROKER Units of work are persistent.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This attribute can be overridden by the STORE field in the Broker ACI control block.						
TRANSLATION	NO | *name* (A255)	O	z	u	w	v	b
	Activates translation user exit for character conversion. NO If translation is not to be used - e.g., for binary payload (broker messages) - either omit the TRANSLATION attribute or specify TRANSLATION=NO. *name* Name of Translation User Exit. See also *Configuring Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation or *Writing Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. The CONVERSION attribute overrides the TRANSLATION attribute when defined for a service; that is, when TRANSLATION and CONVERSION are both defined, TRANSLATION will be ignored.						
UMSG	Alias for MAX-MESSAGES-IN-UOW.						
UOW-DATA-LIFETIME	<u>1D</u> | *n*S | *n*M | *n*H | *n*D	O	z	u	w	v	b
	Defines the default lifetime for units of work for the service. *n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). This attribute is ignored if PSTORE=NO is defined. If the unit of work (UOW) is inactive, that is, not processed within the time limit, it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block.						
UOW-MSGS	Alias for MAX-MESSAGES-IN-UOW.						
UOW-STATUS-LIFETIME	<u>no value</u> | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	v	b
	The value to be added to the UOW-DATA-LIFETIME (lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself. *n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523).						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	*n*D Number of days (max. 24855). The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.						
UWSTATP	0 | *n*	O	z	u	w	v	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-STATUS-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. This attribute is ignored if PSTORE=NO is defined. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.						
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.						
UWTIME	Alias for UOW-DATA-LIFETIME.						

Wildcard Service Definitions

The special names of CLASS = *, SERVER = * and SERVICE = * are allowed in the service-specific and authorization rule-specific sections of the broker attribute file. These are known as "wildcard" service definitions. If this name is present in the attribute file, any service that registers with the broker and does not have its own entry in the attribute file will inherit the attributes that apply to the first wildcard service definition found.

For example, a server that registers with CLASS=ACLASS, SERVER=ASERVER and SERVICE=ASERVICE can inherit attributes from any of the following entries in the attribute file (this list is not necessarily complete):

```
CLASS = *, SERVER = ASERVER, SERVICE = ASERVICE
CLASS = ACLASS, SERVER = *, SERVICE = *
CLASS = *, SERVER = *, SERVICE = *
```

Of course, if there is a set of attributes that are specifically defined for CLASS=ACLASS, SERVER=ASERVER, SERVICE=ASERVICE, then all of the wildcard service definitions will be ignored in favor of the exact matching definition.

Service Update Modes

EntireX has two modes for handling service-specific attributes. See broker-specific attribute SERVICE-UPDATES.

- In **service update mode** (SERVICE-UPDATES=YES), the service configuration sections of the attribute file are read whenever the first replica of a particular service registers.
- In **non-update mode** (SERVICE-UPDATES=NO), the attribute file is not reread. All attributes are read during startup and the broker does not honor any changes in the attribute file. This mode is useful if
  - there is a high frequency of REGISTER operations, or
  - the attribute file is rather large and results in a high I/O rate for the broker.

  The disadvantage to using non-update mode is that if specific attributes are modified, the broker must be restarted to effect the changes. Generally, this mode should be used only if the I/O rate of the broker is considerably high, and if the environment seldom changes.

OPTION Values for Conversion

The different option values allow you to either handle character conversion deficiencies as errors, or to ignore them:

1. Do not ignore any character conversion errors and force an error always (value STOP). This is the default behavior.
2. Ignore if characters cannot be converted into the receiver's codepage, but force an error if sender characters do not match the sender's codepage (value SUBSTITUTE-NONCONV).
3. Ignore any character conversion errors (values SUBSTITUTE and BLANKOUT).

Situations 1 and 2 above are reported to the broker log file if the TRACE option for CONVERSION is set to level 1.

Value	Description	Options Supported for		Report Situation in Broker Log File if TRACE Option for CONVERSION is set to 1	
		SAGTCHA	SAGTRPC	Bad Input Characters (Sender's Codepage)	Non-convertible Characters (Receiver's Codepage)
SUBSTITUTE	Substitutes both non-convertible characters (receiver's codepage) and bad input characters (sender's codepage) with a codepage-dependent default replacement character.	YES	YES	No message.	No message
SUBSTITUTE-NONCONV	If a corresponding code point is not available in the receiver's codepage, the character cannot be converted and is substituted with a codepage-dependent default replacement character. Bad input characters in sender's codepage are not substituted and result in an error.	YES	YES	Write detailed conversion error message.	No message.
BLANKOUT	Substitutes non-convertible characters with a codepage-dependent default replacement; blanks out the complete RPC IDL field containing one or more bad input characters.	NO	YES	No message.	No message.
STOP	Signals an error on detecting a non-convertible or bad input character. This is the default behavior if no option is specified.	YES	YES	Write detailed conversion error message.	Write detailed conversion error message.

Codepage-specific Attributes

The codepage-specific attribute section begins with the keyword `DEFAULTS=CODEPAGE` as shown in the sample attribute file. You can use the attributes in this section to customize the broker's locale string defaults and customize the mapping of locale strings to codepages for character conversion with ICU conversion and SAGTRPC user exit. See *Internationalization with EntireX* for more information.

			Operating System				
Attribute	**Values**	**Opt/ Req**	**z/OS**	**UNIX**	**Windows**	**z/VSE**	**BS2000**
`DEFAULT_ASCII`	Any ICU converter name or alias. See also ***Additional Notes*** below.	O	z	u	w	v	b

	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the broker's locale string defaults if
	■ the calling component does not send a locale string itself, and
	■ the calling component is running on an ASCII platform (UNIX, Windows, etc.)
	Example:
	`DEFAULTS=CODEPAGE`
	`    * Broker Locale String Defaults`
	`    DEFAULT_ASCII=windows-950`
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also ***Additional Notes*** below.						

`DEFAULT_EBCDIC_IBM`	Any ICU converter name or alias	O	z	u	w	v	b
	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the broker's locale string defaults if ■ the calling component does not send a locale string itself and ■ the calling component is running on an IBM mainframe platform (z/OS, z/VSE etc.) Example:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

	`DEFAULT=CODEPAGE`
	`    DEFAULT_EBCDIC_IBM=ibm-937`
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also *Additional Notes* below.						

`DEFAULT_EBCDIC_SNI`	Any ICU converter name or alias.	O	z	u	w	v	b

	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the locale string defaults if
	■ the calling component does not send a locale string itself, and
	■ the calling component is running on a Fujitsu EBCDIC mainframe platform (BS2000)
	Example:
	`DEFAULT=CODEPAGE`
	`    DEFAULT_EBCDIC_SNI= bs2000-edf03drv`
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also *Additional Notes* below.						

`locale-string`	Any ICU converter name or alias. See also *Additional Notes* below.	O	z	u	w	v	
	Customize the mapping of locale strings to codepages and bypass the broker's locale string processing mechanism. See *Broker's Locale String Processing*. This is useful: ■ if the broker's locale string processing fails - i.e. leads to no codepage or to the wrong codepage - you can explicitly assign the codepage which meets your requirements. ■ if you want to install user-written ICU converters (codepages) into the broker, see *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. The attribute (locale string) is the locale string sent by your EntireX component (client or server) and the value is the codepage that you want to use in place of that locale string. In the first line of the example below, the client or server application sends ASCII as a locale string; the broker maps this to the codepage ISO 8859_1. In the same way EUC_JP_LINUX is mapped to ibm-33722_P12A-1999. All other locale strings are mapped by the broker's mapping mechanism, see *Broker's Built-in Locale String Mapping*. Example:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

	`DEFAULTS=CODEPAGE`
	`    * Broker Locale String Codepage Assignments`
	`    ASCII=ISO8859`
	`    EUC_JP_LINUX=ibm-33722_P12A-1999`
	`    * Customer-written ICU converters`
	`    CP1140=myebcdic`
	`    CP0819=myascii`
	For more examples, see *Bypassing Broker's Built-in Locale String Mapping* and also *Additional Notes* below.						

Additional Notes

- Locale string matching is case insensitive when bypassing the broker's built-in mechanism, that is, when the broker examines the codepages section in the attribute file.
- If ICU is used for character conversion and the style in not known by ICU, e.g. <ll>_<cc> etc., the name will be mapped to a suitable ICU alias. For more details on the mapping mechanism, see *Broker's Built-in Locale String Mapping*. For more details on ICU and ICU converter name standards, see *ICU Resources*.
- If SAGTRPC user exit is used for the character conversion, we recommend assigning the codepage in the form CP<*nnnnn*>. To determine the number given to SAGTRPC user exit, see *Broker's Built-in Locale String Mapping*.
- See `CONVERSION` on this page for the character conversion in use.

Adabas SVC/Entire Net-Work-specific Attributes

The Adabas SVC/Entire Net-Work-specific attribute section begins with the keyword `DEFAULTS=NET` as shown in the sample attribute file. The attributes in this section are needed to execute the Adabas SVC/Entire Net-Work communicator of the EntireX Broker kernel.

Note: This section applies to mainframe platforms only. It does not apply to UNIX and Windows.

		Opt/ Req	Operating System				
Attribute	**Values**		**z/OS**	**UNIX**	**Windows**	**z/VSE**	**BS2000**
`ADASVC`	*nnn*	R	z			v	
	Sets the Adabas SVC number for EntireX Broker access. The Adabas SVC is used to perform various internal functions, including communication between the caller program and EntireX Broker. Not supported on BS2000.						
`EXTENDED-ACB-SUPPORT`	`NO` | `YES`	O	z			v	b
	Determines whether extended features of Adabas version 8 (or above) are supported. `NO` No features of Adabas version 8 or above will be used. `YES` Informs broker kernel to provide Adabas/WAL version 8 transport capability. This parameter is required for sending/receiving more than 32 KB data over Adabas [NET] transport. This value should be set only if you have installed Adabas/WAL version 8, Adabas SVC, and included Adabas/WAL version 8 load libraries into the steplib of broker kernel; otherwise, unpredictable results can occur.						
`FORCE`	`NO` | `YES`	O	z			v	b
	Determines whether DBID table entries can be overwritten. `NO` Overwrite of DBID table entries not permitted. `YES` Overwrite of DBID table entries permitted. This is required when the DBID table entry is not deleted after abnormal termination. **Caution:** Overwriting an existing entry prevents any further communication with the overwritten node. Use `FORCE=YES` only if you are absolutely sure that no target node with that DBID is active.						
`IDTNAME`	*idtname*(A8) | `ADABAS5B`	O					b
	If an ID table name is specified with the appropriate `ADARUN` parameter for Entire Net-Work, Adabas or Natural, the same name must be specified here.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	The ID table is used to perform various internal functions, including communication between the caller program and the EntireX Broker. Only supported under BS2000.						
IUBL	8000 | *n*	O	z			v	b
	This parameter sets the maximum length (in bytes) of the buffer that can be passed from the caller to EntireX Broker. The maximum size of IUBL is the same as the maximum value of the Adabas parameter LU (see the *Adabas Operations Manual*). IUBL must be large enough to hold the maximum send-length plus receive-length required for any caller program plus any administrative overhead for Adabas and Entire Net-Work control structures.						
LOCAL	NO | YES	O	z			v	b
	For remote nodes accessed via Entire Net-Work, the attribute LOCAL specifies whether the target ID defined with the NODE attribute can be accessed only locally, or also remotely. NO DBID is *global* and can be accessed from remote nodes via Entire Net-Work. YES DBID is *local* and cannot be accessed from remote nodes via Entire Net-Work.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	v	b
	Maximum message size that the broker kernel can process using transport method NET. The default value represents the highest positive number that can be stored in a four-byte integer.						
NABS	10 | *n*	O	z			v	b
	The number of attached buffers to be used (max. 524287). An attached buffer is an internal buffer used for interprocess communication. An attached buffer pool equal to the NABS value multiplied by 4096 will be allocated. This buffer pool must be large enough to hold all data (IUBL) of all parallel calls to EntireX Broker. The following formula can be used to calculate the value for NABS: `NABS = NCQE *IUBL / 4096.`						
NCQE	10 | *n*	O	z			v	b
	NCQE defines the number of command queue elements which are available for processing commands arriving at the broker kernel over Adabas SVC / Net-Work transport mechanism. Sufficient NCQE should be allocated to allow this transport mechanism to process multiple broker commands concurrently. Each command queue element requires 192 bytes, and the element is released when either the user (client or server) has received the results of the command, or if the command is timed out.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	The number of command queue elements required to handle broker calls depends on the number of parallel active broker calls that are using the transport mechanism Adabas SVC / Entire Net-Work. For example, all broker commands issued by client or server components using this transport mechanism:						
NODE	1-65534	R	z			v	b
	Defines the unique DBID for EntireX Broker. Used for internode Adabas/Entire Net-Work communication. There is no default; the value of NODE must be a value greater than or equal to 1 or less than or equal to 65534. If you set the parameter LOCAL=YES, you can use the same node number for different installations of EntireX Broker in an Entire Net-Work environment.						
TIME	30 | *n*	O	z			v	b
	This parameter sets the timeout value for broker calls in seconds. The results of a broker call must be received by the caller within this time limit.						
TRACE-LEVEL	0-4	O	z			v	b
	The level of tracing to be performed while the broker is running with transport method NET. It overrides the global value of trace level for all NET routines. 0 No tracing. Default value. 1 Display invalid Adabas commands. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						

Security-specific Attributes

The security-specific attribute section begins with the keyword DEFAULTS=SECURITY as shown in the sample attribute file. This section applies only if broker-specific attribute SECURITY=YES is specified.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
ACCESS-SECURITY-SERVER	NO | YES	O					b
	Determines where authentication is checked. NO Authentication is checked in the broker tasks. This requires broker to be running under TSOS in order to execute privileged security checks. YES Authentication is checked in the EntireX Broker Security Server for BS2000. This does not require broker to be running under TSOS. See *EntireX Broker Security Server for BS2000*.						
APPLICATION-NAME	A8	O	z				

	Specifies the name of the application to be checked if FACILITY-CHECK=YES is defined. In RACF, for example, an application BROKER with read permission for user DOE is defined with following commands:
	`RDEFINE APPL BROKER UACC(NONE)`
	`PERMIT BROKER CLASS(APPL) ID(DOE) ACCESS(READ)`
	`SETROPTS CLASSACT(APPL)`
	See attribute FACILITY-CHECK for more information.						

AUTHORIZATION-DEFAULT	YES | NO	O		u	w		
	Determines whether access is granted to a specified service if the specified service could not be found listed in the repository of authorization rules or in section *DEFAULTS=AUTHORIZATION-RULES* of the attribute file. YES Grant access. NO Deny access. Applies only when using EntireX Security under UNIX and Windows. Authorization rules can be stored within a repository. When an authorization call occurs, EntireX Security uses the values of this parameter to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.						
CHECK-IP-ADDRESS	YES | NO	O	z				
	Determines whether the TCP/IP address of the caller is subject to a resource check.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
ERRTXT-MODULE	NA2MSG0 | NA2MSG1 | NA2MSG2 | *ModuleName*	O	z				
	Specifies the name of the security error text module. Default is NA2MSG0, English messages. For instructions on how to customize messages, see *Build Language-specific Messages (Optional)* under *Installing EntireX Security under z/OS*.						
FACILITY-CHECK	NO | YES	O	z				
	It is possible to check whether a particular user is at all allowed to use an application before performing a password check. The advantage of this additional check is that when the user is not allowed to use this application, the broker returns error 00080013 and does not try to authenticate the user. Failing an authentication check may lead to the user's password being revoked; this situation is avoided if the facility check is performed first. See attribute APPLICATION-NAME for further details. **Note:** This facility check is an additional call to the security subsystem and is executed before each authentication call.						
IGNORE-STOKEN	NO | YES	O	z	u	w		b
	Determines whether the value of the ACI field SECURITY-TOKEN is verified on each call.						
INCLUDE-CLASS	YES | NO	O	z				
	Determines whether the class name is included in the resource check.						
INCLUDE-NAME	YES | NO	O	z				
	Determines whether the server name is included in the resource check.						
INCLUDE-SERVICE	YES | NO	O	z				
	Determines whether the service name is included in the resource check.						
LDAP-AUTHENTICATION-URL	*ldapUrl*	O		u	w		

	Authentication is performed against the LDAP repository specified under *ldapUrl*.
	■ **TCP**
	Specify repository URL:
	`LDAP-AUTHENTICATION-URL="ldap://HostName[:PortNumber]"`
	■ **SSL/TLS**
	Specify repository URL with ldaps:
	`LDAP-AUTHENTICATION-URL="ldaps://HostName[:PortNumber]"`
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Examples for TCP and SSL/TLS:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

	`LDAP-AUTHENTICATION-URL="ldap://myhost.mydomain.com"`
	`LDAP-AUTHENTICATION-URL="ldaps://myhost.mydomain.com:636"`						

`LDAP-AUTHORIZATION-URL`	*`ldapUrl`*	O		u	w		

	Authorization is performed against the LDAP repository specified under *`ldapUrl`*.
	■ **TCP**
	Specify repository URL:
	`LDAP-AUTHORIZATION-URL="ldap://HostName[:PortNumber]"`
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport.
	Example for TCP:
	`LDAP-AUTHORIZATION-URL="ldap://myhost.mydomain.com:389"`
	This attribute replaces the parameters `host`, `port` and `protocol` in the *xds.ini* file of EntireX version 9.10 and below.						

`LDAP-AUTH-DN`	*`authDN`*	O		u	w		
	For authenticated access to the LDAP server. Specifies the DN of the user. Default value: `cn=admin,dc=software-ag,dc=de` This attribute replaces parameter `authDN` in the *xds.ini* file of EntireX version 9.10 and below.						
`LDAP-AUTH-PASSWD-ENCRYPTED`	*`authPass`*	O		u	w		
	For authenticated access to the LDAP server. Specifies the encrypted value of the user password. Use program `etbnattr` to get the encrypted password: `etbnattr -x clear_text_password -echo_password_only` This writes the encrypted password to standard output. This attribute replaces parameter `authPass` in the *xds.ini* file of EntireX version 9.10 and below.						
`LDAP-AUTHORIZATION-RULE`	`A32`	O		u	w		
	List of authorization rules. Multiple sets of rules can be defined, each set is limited to 32 chars. The maximum number of `LDAP-AUTHORIZATION-RULE` entries in the attribute file is 16. Applies only when using EntireX Security under UNIX or Windows and `SECURITY-SYSTEM=`*`ldapUrl`*. Authorization rules can be stored in an LDAP repository. When an authorization call occurs, EntireX Security uses the values of this parameter						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	and AUTHORIZATION-DEFAULT to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.						
LDAP-BASE-DN	*baseDN*	O		u	w		
	Specifies the base distinguished name of the directory object that is the root of all objects for authorization rules. Default value: `dc=software-ag,dc=de` This attribute replaces parameter baseDN in the *xds.ini* file of EntireX version 9.10 and below.						
LDAP-PERSON-BASE-BINDDN	*ldapDn*	O		u	w		
	Used with LDAP authentication to specify the distinguished name where authentication information is stored. This value is prefixed with the user ID field name (see below). Example: `LDAP-PERSON-BASE-BINDDN="cn=users,dc=mydomain,dc=com"`						
LDAP-REPOSITORY-TYPE	OpenLDAP | ActiveDirectory | SunOneDirectory | Tivoli | Novell | ApacheDS	O		u	w		
	Use predefined known fields for the respective repository type. Specify the repository type that most closely matches your actual repository. In the case of Windows Active Directory, the user ID is typically in the form *domainName*\userId.						
LDAP-SASL-AUTHENTICATION	NO | YES	O			w		
	Specifies whether or not Simple Authentication and Security Layer (SASL) is to perform the authentication check. In practice, this determines whether or not the password supplied by the user is passed in plain text between the broker kernel and the LDAP server. If SASL is activated, this implies that the password is encrypted. NO Password is sent to LDAP server in plain text. YES Password is sent to LDAP server encrypted.						
LDAP-USERID-FIELD	cn | *uidFieldName*	O		u	w		
	Used with LDAP authentication to specify the first field name of a user in the Distinguished Name, for example: LDAP-USERID-FIELD=*uid*						
MAX-SAF-PROF-LENGTH	1-256	O	z				
	This parameter should be increased if the length of the resource checks - that is, the length of the profile comprising "<class>.<server>.<service>" - is greater than 80 bytes.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This parameter defaults to 80 if a value is not specified.						
PASSWORD-TO-UPPER-CASE	NO | YES	O	z			v	
	Determines whether the password and new password are converted to uppercase before verification.						
PRODUCT	RACF | ACF2 | TOP-SECRET	O	z				
	Specifies the name of the installed security product. This attribute is used to analyze security-system-specific errors. The following systems are currently supported: ACF2 Security system ACF2 is installed. RACF Security system RACF is installed. Default. TOP-SECRET Security system TOP-SECRET is installed. The default value is used if an incorrect or no value is specified.						
PROPAGATE-TRUSTED-USERID	YES | NO	O	z				
	Determines whether a client user ID obtained by means of the trusted user ID mechanism is propagated to a server using the ACI field CLIENT-USERID.						
SAF-CLASS	NBKSAG | *SAFClassName*	O	z				
	Specifies the name of the SAF class/type used to hold the EntireX-related resource profiles.						
SAF-CLASS-IP	NBKSAG | *SAFClassName*	O	z				
	Specifies the name of the SAF class/type used when performing IP address authorization checks.						
SECURITY-LEVEL	AUTHORIZATION | AUTHENTICATION	O	z	u	w	v	b
	Specifies the mode of operation. AUTHORIZATION Authorization and authentication (not under BS2000 or z/VSE). AUTHENTICATION Authentication. **Note:** In version 8.0, the default value for this parameter was AUTHORIZATION.						
SECURITY-NODE	YES | *name*	O	z				
	This parameter can be used to specify a prefix that is added to all authorization checks, enabling different broker kernels, in different environments, to perform separate authorization checks according to each broker kernel. For example, it is often important to distinguish between production, test, and development environments. YES This causes the broker ID to be used as a prefix for all authorization checks.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	name This causes the actual text (maximum 8 characters) to be prefixed onto all authorization checks. **Note:** By *not* setting this parameter, no prefix is added to the resource check (the default behavior).						
SECURITY-SYSTEM	OS | LDAP	O	z	u	w		b
	OS Authentication is performed against the local operating system. Default if SECURITY=YES is specified and section DEFAULTS=SECURITY is omitted from the attribute file. LDAP Authentication and authorization are performed against the LDAP repository specified under LDAP-AUTHENTICATION-URL and LDAP-AUTHORIZATION-URL.						
TRACE-LEVEL	0-4	O	z	u	w	v	b
	Trace level for EntireX Security. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log security violations and access denied/permitted. 2 All of trace level 1, plus internal errors. 3 All of trace level 2, plus function entered/exit messages with argument values and some progress messages. 4 All of trace level 3, plus some selected data areas for problem analysis. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD. **Note:** Setting this value also affects tracing for authorization rules.						
TRUSTED-USERID	YES | NO	O	z				
	Activates the trusted user ID mechanism for broker requests arriving over the local Adabas IPC mechanism.						
USERID-TO-UPPER-CASE	NO | YES	O	z			v	
	Determines whether user ID is converted to uppercase before verification.						
UNIVERSAL	NO | YES	O	z				
	Determines whether access to undefined resource profiles is allowed.						
WARN-MODE	NO | YES	O	z	u	w		b
	Determines whether a resource check failure results in just a warning or an error.						

TCP/IP-specific Attributes

The TCP/IP-specific attribute section begins with the keyword DEFAULTS=TCP as shown in the sample attribute file. It contains attributes that apply to the TCP/IP transport communicator. The transport is activated by TRANSPORT=TCP in the Broker-specific section of the attribute file. A maximum of five TCP/IP communicators can be activated by specifying up to five HOST/PORT pairs.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O	z	u	w	v	b
	Non-activity of the TCP/IP connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity test is disabled. On the stub side, non-activity can be set with the environment variable ETB_NONACT. See *Limiting the TCP/IP Connection Lifetime* in the platform-specific *Stub Administration* sections of the EntireX documentation.						
HOST	0.0.0.0 | *HostName* | *IP address*	O	z	u	w	v	b
	The address of the network interface on which broker will listen for connection requests. If HOST is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five HOST/PORT pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	v	b
	Maximum message size that the broker kernel can process using transport method TCP/IP. The default value represents the highest positive number that can be stored in a four-byte integer.						
PORT	1025-65535	O	z	u	w	v	b

<table>
<tr><th rowspan="2">Attribute</th><th rowspan="2">Values</th><th rowspan="2">Opt/ Req</th><th colspan="5">Operating System</th></tr>
<tr><th>z/OS</th><th>UNIX</th><th>Windows</th><th>z/VSE</th><th>BS2000</th></tr>
<tr><td></td><td colspan="6">The TCP/IP port number on which the broker will listen for connection requests.
If not specified, the broker will attempt to find its TCP/IP port number from the TCP/IP services file, using getservbyname. If it cannot find the number here, the default value of 1971 is used.
A maximum of five HOST/PORT pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.
Example for multiple ports on z/OS:
HOST=localhost,PORT=3930
HOST=0.0.0.0,PORT=3931
- Port 3930 is used for *local* TCP/IP communication only and is not visible outside the z/OS host.
- Port 3931 is used for *global* TCP/IP communication. With IBM's AT-TLS this port is turned into a TLS port, see *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
With this configuration you can reach the broker from outside the z/OS host via the secure TLS connection only (port 3931). The TCP connection (port 3930) can only be used from inside the z/OS host.</td><td></td></tr>
<tr><td rowspan="2">RESTART</td><td>YES | NO</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">YES The broker kernel will attempt to restart the TCP/IP communicator.
NO The broker kernel will not try to restart the TCP/IP communicator.
This setting applies to all TCP/IP communicators.</td></tr>
<tr><td rowspan="2">RETRY-LIMIT</td><td>20 | *n* | UNLIM</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Maximum number of attempts to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.</td></tr>
<tr><td rowspan="2">RETRY-TIME</td><td>3M | *n* | *n*S | *n*M | *n*H</td><td>O</td><td>z</td><td>u</td><td>w</td><td>v</td><td>b</td></tr>
<tr><td colspan="7">Wait time between stopping the TCP/IP communicator due to an unrecoverable error and the next attempt to restart it.
n Same as *n*S.
*n*S Wait time in seconds (max. 2147483647).
*n*M Wait time in minutes (max. 35791394).
*n*H Wait time in hours (max. 596523).
Minimum wait time is 1S.</td></tr>
</table>

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This setting applies to all TCP/IP communicators.						
REUSE-ADDRESS	YES | NO	O	z	u		v	b
	YES | NO	O			w		
	YES The TCP port assigned to the broker can be taken over and assigned to other applications (this is the default value on all non-Windows platforms). NO The TCP port assigned to the broker cannot be taken over and assigned to other applications. This is the default setting on Windows, and we strongly advise you do not change this value on this platform. **Note:** This setting might be required at your site when restarting broker immediately after stopping it. This is due to the inherent latency of the TCP/IP stack when closing connections.						
STACK-NAME	*StackName*	O	z				
	Name of the TCP/IP stack that the broker is using. If not specified, broker will connect to the default TCP/IP stack running on the machine.						
TRACE-LEVEL	0-4	O	z	u	w	v	b
	The level of tracing to be performed while the broker is running with transport method TCP/IP. It overrides the global value of trace level for all TCP/IP routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						

c-tree-specific Attributes

The c-tree-specific attribute section begins with the keyword `DEFAULTS = CTREE`. The attributes in this section are optional. This section applies only if `PSTORE-TYPE = CTREE` is specified.

Not available under z/OS, BS2000, z/VSE.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
COMPATIBILITY	NO | YES	O		u	w		
	Determines whether the following c-tree parameters are set: - COMPATIBILITY PREV610A_FLUSH - COMPATIBILITY FDATASYNC - SUPPRESS_LOG_FLUSH YES - PREIMAGE_DUMP YES See your FairCom documentation for a description of these parameters. NO The c-tree parameters listed above are not set. Default. YES The c-tree parameters listed above are set. This provides compatibility with c-tree behavior prior to EntireX Broker 10.5.						
FLUSH-DIR	YES | NO	O		u	w		
	Controls whether metadata is flushed to disk immediately after creates, renames, and deletes of transaction log files and transaction-dependent files. YES Metadata is flushed to disk. NO Metadata is not flushed to disk. This provides compatibility with c-tree behavior prior to EntireX Broker version 10.5. See `COMPATIBILITY NO_FLUSH_DIR` in the FairCom documentation for a description of this parameter.						
MAXSIZE	*n* | *n*M | *n*G	O		u	w		
	Defines the maximum size of c-tree data files. Broker allocates one data file for control data and another data file for message data: *n* Maximum size in MB. *n*M Maximum size in MB. *n*G Maximum size in GB.						
PAGESIZE	*n* | *n*K	O		u	w		
	Determines how many bytes are available in each c-tree node. `PSTORE COLD` start is required after changing this value.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	n Same as *n*K *n*K PAGESIZE in KB. The default and minimum value is 8 KB. If PSD Reason Code = 527 is returned during UOW write processing, increase the PAGESIZE value and restart broker with PSTORE=COLD, or migrate the existing PSTORE to a new PSTORE with an increased PAGESIZE value. See *Migrating the Persistent Store* and define the increased PAGESIZE value for the load broker.						
PATH	A255	O		u	w		
	Path name of the target directory for c-tree index and data files.						
SYNCIO	NO | YES	O		u	w		
	Controls the open mode of the c-tree transaction log. NO c-tree transaction log is not opened in synchronous mode. Default. YES c-tree transaction log is opened in synchronous mode to improve data security. It may degrade performance of PSTORE operations, but offers the highest level of data security. See *c-tree Database as Persistent Store* in the UNIX and Windows Administration documentation.						
TRACE-LEVEL	0-4	O		u	w		
	Trace level for c-tree persistent store. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log memory allocation failures and errors during close of files. 2 n/a 3 All of trace level 1, plus UOWID in use for the various ctree requests and function entered/exit mesages. 4 All of trace level 3, plus returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						

SSL/TLS-specific Attributes

The Broker 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 clients and servers, as well as ACI clients and servers, are always SSL clients. The broker is always the SSL server. For an introduction see *SSL/TLS and Certificates with EntireX*.

Your operating system and, for z/OS, the approach you use determine whether this section of the attribute file is required:

- **z/OS**
  - **AT-TLS**
    This is the approach we recommend. IBM's Application Transparent Transport Layer Security (AT-TLS) does not require the SSL-specific attribute section.
  - **Direct SSL**
    For direct SSL/TLS support, the SSL-specific attribute section is required. It begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.

    Note: Direct SSL/TLS support (using GSK) inside the Broker under z/OS will be dropped in the next version. We strongly recommend using IBM's Application Transparent Transport Layer Security (AT-TLS) instead.

  See *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
- **UNIX and Windows**
  The SSL-specific attribute section is required, and begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.

  The attributes in this section are needed to execute the SSL communicator of the EntireX Broker kernel.

  See *Running Broker with SSL/TLS Transport*.
- **z/VSE**
  The SSL-specific attribute section is not used. You can use BSI's Automatic Transport Layer Security (ATLS). See *Running Broker with SSL/TLS Transport* in the z/VSE Administration documentation.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
`CIPHER-SUITE`	*string*	O	z	u	w		b

	String that is passed to the underlying SSL/TLS implementation. SSL/TLS is a standardized protocol that uses different cryptographic functions (hash functions, symmetric and asymmetric encryption etc.). Some of these must be implemented in the SSL/TLS stack; others are optional. When an SSL/TLS connection is created, both parties agree by "handshake" on the cipher suite, that is, the algorithms and key lengths used. In a default scenario, this information depends on what both sides are capable of. It can be influenced by setting the attribute `CIPHER-SUITE` for the SSL/TLS server side (the broker always implements the server side). Thus stubs connect to the broker and thereby become the SSL/TLS clients.
	Under UNIX, Windows and BS2000, the OpenSSL implementation is used; under z/OS it is GSK.
	The SSL protocol is obsolete. It is no longer available. The TLS protocol is the successor of SSL and is readily available in OpenSSL and GSK. The following examples show how to configure the available cipher suites:
	■ **OpenSSL**
	The default configuration uses FIPS 140-2 approved cipher suites, eligible for TLS v1.2, but without anonymous Diffie-Hellman (ADH) and pre-shared key (PSK) algorithms. The resulting set of cipher suites provides for authentication and strong encryption:
	`CIPHER-SUITE=FIPS+TLSv1.2:!ADH:!PSK:@STRENGTH`
	See ***https://www.openssl.org/docs/man1.1.1/man1/ciphers***.
	■ **GSK**
	Default configuration:
	`CIPHER-SUITE=9F9D9E9C6B673D3C39383332352F`
	This list of FIPS 140-2 approved cipher suites starts with a strong '256-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate' (9F) and ends with a relatively weak '128-bit AES encryption with SHA-1 message authentication and RSA key exchange' (2F).
	See the IBM documentation *z/OS V2R3 Cryptographic Services System Secure Sockets Layer Programming*, SC14-7495-30, Appendix C: *Cipher Suite Definitions*.						

`CONNECTION-NONACT`	*n* | *n*S | *n*M | *n*H	O	z	u	w		b
	Non-activity of the SSL connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	n Same as *n*`S`. *n*`S` Non-activity time in seconds (min. 600, max. 2147483647). *n*`M` Non-activity time in minutes (min. 10, max. 35791394). *n*`H` Non-activity time in hours (max. 596523). If not specified, the connection non-activity test is disabled.						
`HOST`	*hostname*	O	z	u	w		b
	The address of the network interface on which broker will listen for connection requests. If `HOST` is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five `HOST`/`PORT` pairs can be specified to start multiple instances of EntireX Broker's TCP/IP transport communicator.						
`KEY-LABEL`	*name*	O	z				
	The label of the key in the RACF keyring that is used to authenticate the broker kernel (see also `TRUST-STORE` parameter). Example: `ETBCERT`.						
`KEY-FILE`	*filename*	R		u	w		b
	File that contains the broker's private key (if not contained in `KEY-STORE`). For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: `MyAppKey.pem`. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).						
`KEY-PASSWD`	*password* (A32)	R		u	w		b
	Password used to protect the private key. Unlocks the `KEY-FILE`, for example `MyAppKey.pem`. Deprecated. See `KEY-PASSWD-ENCRYTPED` below.						
`KEY-PASSWD-ENCRYPTED`	*encrypted value* (A64)	R		u	w		b
	Password used to protect the private key. Unlocks the `KEY-FILE`, for example `MyAppKey.pem`. This attribute replaces `KEY-PASSWD` to avoid a clear-text password as attribute value. If `KEY-PASSWD` and `KEY-PASSWD-ENCRYTPED` are both supplied, `KEY-PASSWD-ENCRYTPED` takes precedence. Use program `etbnattr` to get the encrypted password:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	`etbnattr -w ssl_key_password --echo_password_only` This writes the encrypted password to standard output.						
KEY-STORE	*file name*	R		u	w		b
	SSL certificate; may contain the private key. For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: *ExxAppCert.pem*. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w		b
	Maximum message size that the broker kernel can process using transport method SSL. The default value represents the highest positive number that can be stored in a four-byte integer.						
PORT	1025-65535	O	z	u	w		b
	The SSL port number on which the broker will listen for connection requests. If not changed, this parameter takes the standard value as specified in the example attribute file. If the port number is not specified, the broker will use the default value of 1958.						
RESTART	YES | NO	O	z	u	w		b
	YES The broker kernel will attempt to restart the SSL communicator (this is the default value). NO The broker kernel will not attempt to restart the SSL communicator.						
RETRY-LIMIT	20 | *n* | UNLIM	O	z	u	w		b
	Maximum number of attempts to restart the SSL communicator.						
RETRY-TIME	3M | *n* | *n*S | *n*M| *n*H	O	z	u	w		b
	Wait time between suspending SSL communication due to unrecoverable error and the next attempt to restart it. *n* Same as *n*S. *n*S Wait time in seconds (max.2147483647). *n*M Wait time in minutes (max. 35791394). *n*H Wait time in hours (max. 596523). Minimum: 1S						
REUSE-ADDRESS	YES | NO	O	z	u	w		b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	YES The SSL port assigned to the broker can be taken over and assigned to other applications (this is the default value). NO The SSL port assigned to the broker cannot be taken over and assigned to other applications. **Note:** This setting might be required at your site when restarting broker immediately after stopping it. This is due to the inherent latency of the TCP/IP stack when closing connections.						
STACK-NAME	*name*	O	z	u	w		
	Name of the TCP/IP stack that the broker is using. If not specified, broker will connect to the default TCP/IP stack running on the machine.						
TRACE-LEVEL	0-4	O	z	u	w		b
	The level of tracing to be performed while the broker is running with transport method SSL/TLS. It overrides the global value of trace level for all SSL/TLS routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						
TRUST-STORE	*file name* | *keyring*	R	z	u	w		b
	Location of the store containing certificates of trust Certificate Authorities (or CAs). ■ **z/OS** Specify the RACF keyring using the following format: [*USER-ID/*]*RING-NAME*. If no value for USER-ID is provided, the keyring is assumed to be associated with the user ID that the broker kernel is running under.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	■ **UNIX/Windows/BS2000** Specify the file name of the CA certificate store. Examples: `EXXCACERT.PEM`, `C:\Certs\ExxCACert.pem`						
`VERIFY-CLIENT`	<u>`NO`</u> | `YES`	O	z	u	w		b
	`YES` Additional client certificate required. `NO` No client certificate required (default).						

DIV-specific Attributes

These attributes define a persistent store that is implemented as a VSAM linear data set (LDS) accessed using Data In Virtual (DIV). This DIV persistent store is a container for units of work. The DIV-specific attribute section begins with the keyword `DEFAULTS = DIV`. The attributes in this section are required if `PSTORE-TYPE = DIV` is specified.

Note: All attributes except the deprecated `DIV` were introduced with EntireX version 9.12. They replace the *Format Parameters* of earlier versions, which are deprecated but still supported for compatibility reasons.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
`DIV`	`A511`	O	z				
	The VSAM persistent store parameters, enclosed in double quotes (""). The value can span more than one line. **Note:** Deprecated. This attribute is applicable only if you are supplying the persistent store parameters using *Format Parameters* of earlier versions. We recommend you use the attributes below that were introduced with EntireX 9.12 instead.						
`DATASPACE-NAME`	`A8`	O	z				
	Defines the name of the dataspace that will be used to map the persistent store. Default value is `DSPSTORE`.						
`DATASPACE-PAGES`	`126-524284`	O	z				
	Defines the size of the dataspace used to map the persistent store (size=`DATASPACE-PAGES` * 4 KB). We recommend using the maximum value. Default value is 2048.						
`DDNAME`	`A8`	R	z				
	Defines the JCL `DDNAME` that will be used to access the persistent store.						
`STORE`	`A8`	R	z				
	Defines an internal name that is used to identify the persistent store.						
`TRACE-LEVEL`	`0-4`	O	z				
	Trace level for DIV. It overrides the global value of trace level in the attribute file. `0` No tracing. Default value. `1` Log selected `DIV SAVE` calls taking longer than 2 seconds elapsed time. `2` n/a `3` All of trace level 1, plus UOWID in use for the various DIV requests. `4` n/a						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						

Adabas-specific Attributes

The Adabas-specific attribute section begins with the keyword DEFAULTS = ADABAS. The attributes in this section are required if PSTORE-TYPE = ADABAS is specified. In previous versions of EntireX, these Adabas-specific attributes and values were specified in the broker-specific PSTORE-TYPE attribute.

Attribute	Values	Opt/Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
BLKSIZE	126-20000	O	z	u	w	v	b
	Optional blocking factor used for message data. If not specified, broker will split the message data into 2 KB blocks to be stored in Adabas records. The maximum value depends on the physical device assigned to data storage. See the *Adabas* documentation. For reasons of efficiency, do not specify a BLKSIZE much larger than the actual total size of the UOW data to be written. The total UOW size is the sum of all messages in the UOW plus 41 bytes of header information. This takes effect only after COLD start. The BLKSIZE parameter applies only for a cold start of broker; subsequently the value of BLKSIZE is taken from the last cold start. Default value is 2000.						
DBID	1-32535	R	z	u	w	v	b
	Database ID of Adabas database where the persistent store resides.						
FNR	1-32535	R	z	u	w	v	b
	File number of broker persistent store file.						
FORCE-COLD	N | Y	O	z	u	w	v	b
	Determines whether a broker cold start is permitted to overwrite a persistent store file that has been used by another broker ID and/or platform. Specify Y to allow existing information to be overwritten.						
MAXSCAN	0-*n*	O	z	u	w	v	b
	Limits display of persistent UOW information in the persistent store through Command and Information Services. Default value is 1000.						
OPENRQ	N | Y	O	z	u	w	v	b
	Determines whether driver for Adabas persistent store is to issue an OPEN command to Adabas.						
SVC	200-255	R	z			v	
	Use this parameter to specify the Adabas SVC number to be used by the Adabas persistent store driver.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
TRACE-LEVEL	0-4	O	z	u	w	v	b
	Trace level for Adabas persistent store. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log selected Adabas CB fields (command code, response code, subcode, ISN, additions). 2 n/a 3 All of trace level 1, plus UOWID in use for the various Adabas requests and function entered/exit mesages. 4 All of trace level 3, plus more Adabas CB fields for successful requests and returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.						

Application Monitoring-specific Attributes

The application monitoring-specific attribute section begins with the keyword `DEFAULTS=APPLICATION-MONITORING`. It contains attributes that apply to the application monitoring functionality. At startup time, the attributes are read if the Broker-specific attribute `APPLICATION-MONITORING=YES` is specified. Duplicate or missing values are treated as errors. When an error occurs, application monitoring is turned off and EntireX Broker continues execution. See *Application Monitoring*.

Attribute	Values	Opt/Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
`APPLICATION-MONITORING-NAME` or `APPMON-NAME`	`A100`	O	z	u	w	v	b
	Specifies a default application monitoring name. Used to set the value of the **ApplicationName** KPI.						
`COLLECTOR-BROKER-ID`	`A64`	R	z	u	w	v	b
	Identifies the Application Monitoring Data Collector. Has the format *`host_name`*:*`port_number`*, where *`host_name`* is the host where the Application Monitoring Data Collector is running and *`port_number`* is the port number of the Application Monitoring Data Collector. The default port is 57900.						
`TRACE-LEVEL`	`0-4`	O	z	u	w	v	b
	The level of tracing to be performed while the broker is running with application monitoring. `0` No tracing. Default value. `1` Display application monitoring errors. `2` All of trace level 1, plus measuring points for application monitoring. `3` All of trace level 2, plus function entered/exit messages with argument values and monitoring buffers. `4` All of trace level 3, plus returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the `TRACE-LEVEL` attribute, you must restart the broker for the change to take effect. `TRACE-LEVEL` cannot be changed dynamically for application monitoring.						

Authorization Rule-specific Attributes

The authorization rule-specific attribute section begins with the keyword DEFAULTS=AUTHORIZATION-RULES. It contains attributes that enhance security-related definitions. At startup time, the attributes are read if the following conditions are met:

- Broker-specific attribute SECURITY=YES
- Security-specific attributes SECURITY-SYSTEM=OS and SECURITY-LEVEL=AUTHORIZATION

When an error occurs, the EntireX Broker stops. See *Authorization Rules*.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
RULE-NAME	A32	R		u	w		
	Specifies a rule name. A rule is a container for a list of services and a list of client and server user IDs. All users defined in a rule are authorized to use all services defined in this rule. See example under *Rules Stored in Broker Attribute File*.						
CLASS SERVER SERVICE	A32	R		u	w		
	These three attributes together identify the service. CLASS must be specified first, followed immediately by SERVER and SERVICE. ***Wildcard Service Definitions*** are allowed.						
CLIENT-USER-ID	A32	R		u	w		
	Defines an authorized client user ID.						
SERVER-USER-ID	A32	R		u	w		
	Defines an authorized server user ID.						

Variable Definition File

The broker attribute file contains the configuration of one EntireX Broker instance. In order to share attribute files between different brokers, you identify the attributes that are unique and move them to a variable definition file. This file enables you to share one attribute file among different brokers. Each broker in such a scenario requires its own variable definition file.

The following attributes are considered unique for each machine:

- `BROKER-ID` (in ***Broker-specific Attributes***)
- `NODE` (in ***Adabas SVC/Entire Net-Work-specific Attributes***)
- `PORT` (in ***SSL/TLS-specific Attributes*** and ***TCP/IP-specific Attributes***)

How you use the variable definition file will depend upon your particular needs. For instance, some optional attributes may require uniqueness - for example, `DBID` and `FNR` in `DEFAULTS=ADABAS` - so that you may specify the persistent store.

6 Concepts of Persistent Messaging

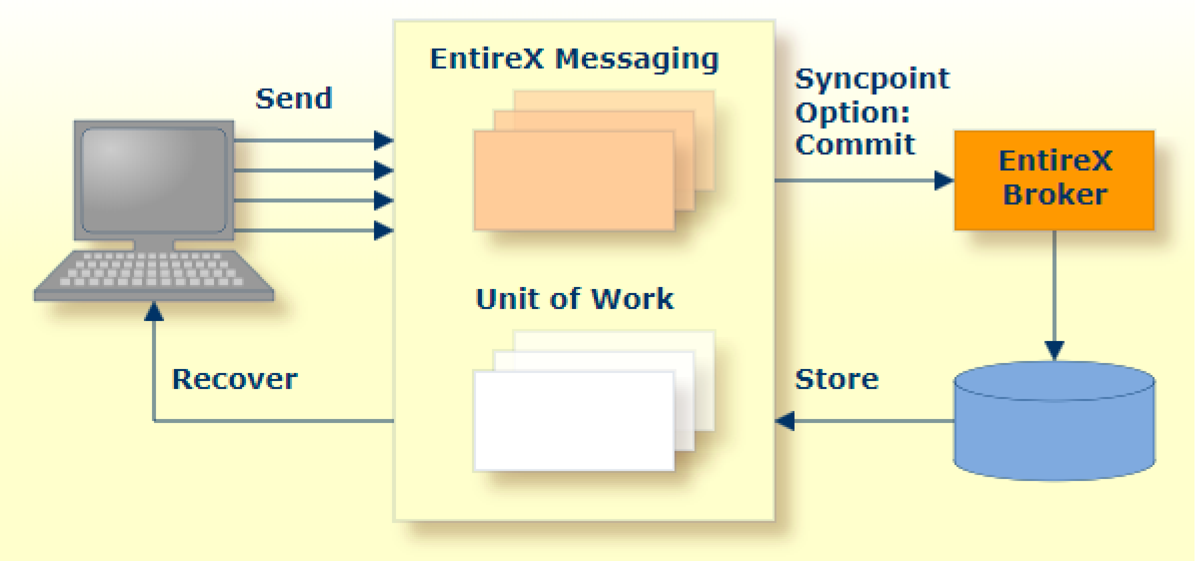
This chapter provides a brief introduction to the concepts of the persistent store and its role in EntireX for providing persistent messaging within the client/server model. It covers the following topics:

The table *Persistent Store Drivers* lists the implementation choices available to each operating system for accessing the physical persistent store. See also *Using Persistence and Units of Work*, *Broker UOW Status Transition* and *Managing the Broker Persistent Store* in the platform-specific Administration documentation.

Client Server Model: Persistent Messaging

EntireX provides persistent messaging within the client/server model. This is achieved by storing all persistent messages on disk so that if a system failure occurs, messages will automatically be recovered allowing applications to be restarted without any loss of data. The section *Using Persistence and Units of Work* describes implementation issues and how to use persistence and units of work in EntireX Broker. Units of work can also be used without persistence; units of work which are the vehicle for persistent messaging.

The following figure illustrates the concept of persistent messages.



Persistence in an EntireX Broker's unit of work (a group of logically related messages) has the following four variations:

- Both the unit of work and its status have persistence.
- The unit of work does not have persistence, but its status does.
- The unit of work has persistence, but its status does not.

- Neither the unit of work nor its status has persistence.

The status of a message is information about the message rather than the actual message data itself. This enables the sender to determine the progress of the message and determine if it has been received by the partner and whether processing was successfully completed. This gives applications the option of having the Broker kernel store only the message status and not the message itself, provided the application has been written to resend data from a known point in the event of system failure. This option can afford significant performance benefits over storing the whole message data.

To support transaction control in a coordinated operation of distributed systems, EntireX can group logically related messages into “units of work” that are committed to the EntireX Broker for further transmission when complete. In case of failure on the server side, the receiving program can backout the whole unit of work; this makes it available for processing later or by another server.

Definitions of Persistent Messaging Terms

UOW

A unit of work (UOW) is a set of one or more messages that are processed as a single unit. The sender of a UOW adds messages to the UOW and then indicates that the UOW is complete (`COMMIT`). The UOW and its messages are not visible to the receiver until the sender has committed the UOW. Once the UOW is committed, the receiver can receive the messages, and can indicate when the UOW is complete (`COMMIT`).

Persistent Store

The persistent store is used for storing unit-of-work messages to disk. This means message and status information can be recovered after a hardware or software failure to the previous commit point issued by each application component.

Persistent Store Drivers

A persistent store driver is an executable, or a load module, that implements access to the physical persistent store. There is one persistent store driver for each persistent store type. The following table shows the persistent store options:

Persistent Store Type	Description	Operating System	Notes
Adabas	Uses Adabas database.	UNIX, Windows, z/OS, BS2000, z/VSE	Adabas, Software AG's ADAptable dataBASe, is a high-performance, multithreaded, database management system.
DIV	Uses IBM Data In Virtual facility on z/OS.	z/OS	This persistent store option is implemented as a VSAM linear data set.
CTREE	c-tree© is an embedded local database that can be used as your persistent store.	UNIX and Windows	c-tree© is the fast and reliable embedded database of FairCom Corporation®.

See also *Managing the Broker Persistent Store* in the platform-specific Administration documentation and also `PSTORE-TYPE` under *Broker Attributes*.

UOW Lifetime

Each UOW has a lifetime value associated with it. This is the period of time that the UOW is allowed to exist without being completed. This time starts when the UOW is initially created and runs until the UOW is completed. A UOW is completed when it is successfully:

- cancelled or backed out by its sender, or
- cancelled or committed by its receiver.

If the UOW is in `ACCEPTED` status when this lifetime expires, the UOW is placed into a `TIMEOUT` status. Lifetime timeouts will not occur when the UOW is in either `RECEIVED` or `DELIVERED` status.

A special "pseudo-clock" is maintained for UOW lifetimes. This clock is implemented in such a way that it only runs when the Broker is active. This prevents a UOW lifetime from expiring while the Broker is down or otherwise unavailable.

Persistent UOW

Persistence is an attribute of a UOW (unit of work). If a UOW is persistent, its messages are saved in the persistent store when the sender `COMMIT`s the UOW and they are retained until the receiver `COMMIT`s or `CANCEL`s the UOW, or until its lifetime expires. If the Broker or system should fail after the UOW is committed by the sender, the UOW (and its conversation) will be restored to their last, stable status when the Broker restarts.

Persistent Status

Persistent status is an attribute of a UOW (unit of work). If a UOW has persistent status, the status of the UOW is maintained in the persistent store, and is updated whenever the status changes. The persistent status remains in the persistent store after the UOW is completed, until its status lifetime has expired.

A persistent status value represents a multiple of the UOW lifetime value. Thus if a UOW has a lifetime of 5M (whereby M stands for minutes) and a persistent status value of 4, the status of the UOW would be deleted 20M (5M*4) after the UOW was completed.

Availability of Persistent Store

Caution: The persistent store must be available before you attempt to start or restart the Broker; otherwise your Broker will not initialize.

Introduction

The PSTORE must be available for the Broker to start. Subsequently, Broker will continue to function even if the PSTORE becomes unavailable and applications issuing non-persistent commands will continue without interruption. However, Broker will not be able to process commands relating to persistence until the PSTORE becomes available again.

Users issuing commands involving persistence - for example units of work with persistence - are notified of the unavailability of the PSTORE through an ACI return code. In addition, persistent commands being processed at the point of unavailability are backed out, and details of the PSTORE problem are written to the Broker log file.

There are several reasons for the PSTORE becoming unavailable. Examples:

- unavailability of the PSTORE file(s)
- capacity of PSTORE file being exceeded
- in the case of Adabas, termination of the database

Disconnect the Persistent Store

You can remove the state “No new Units of Work” - that is, no new persistent data - using CIS. If the PSTORE capacity is exceeded, an error message is written to the Broker log file. You must use Command and Information Services (CIS) to ensure that users cannot issue further commands creating new units of work.

During the time the PSTORE is unavailable, there is no timeout processing for unit-of-work records kept in the PSTORE. In addition, some in-memory resources relating to persistent items, such as conversation control blocks, are also retained until the PSTORE becomes available again and normal processing is resumed for all commands.

See executable command request DISCONNECT-PSTORE under ETBCMD: Executable Command Requests.

Connect the Persistent Store

Subsequently, you can use CIS to make the PSTORE available again, allowing users only to issue commands consuming records from the PSTORE rather than producing new ones. After a period of operation in this state, the contents of the PSTORE will be reduced sufficiently, and you can remove the state "No new Units of Work" through CIS.

See executable command request CONNECT-PSTORE under ETBCMD: Executable Command Requests.

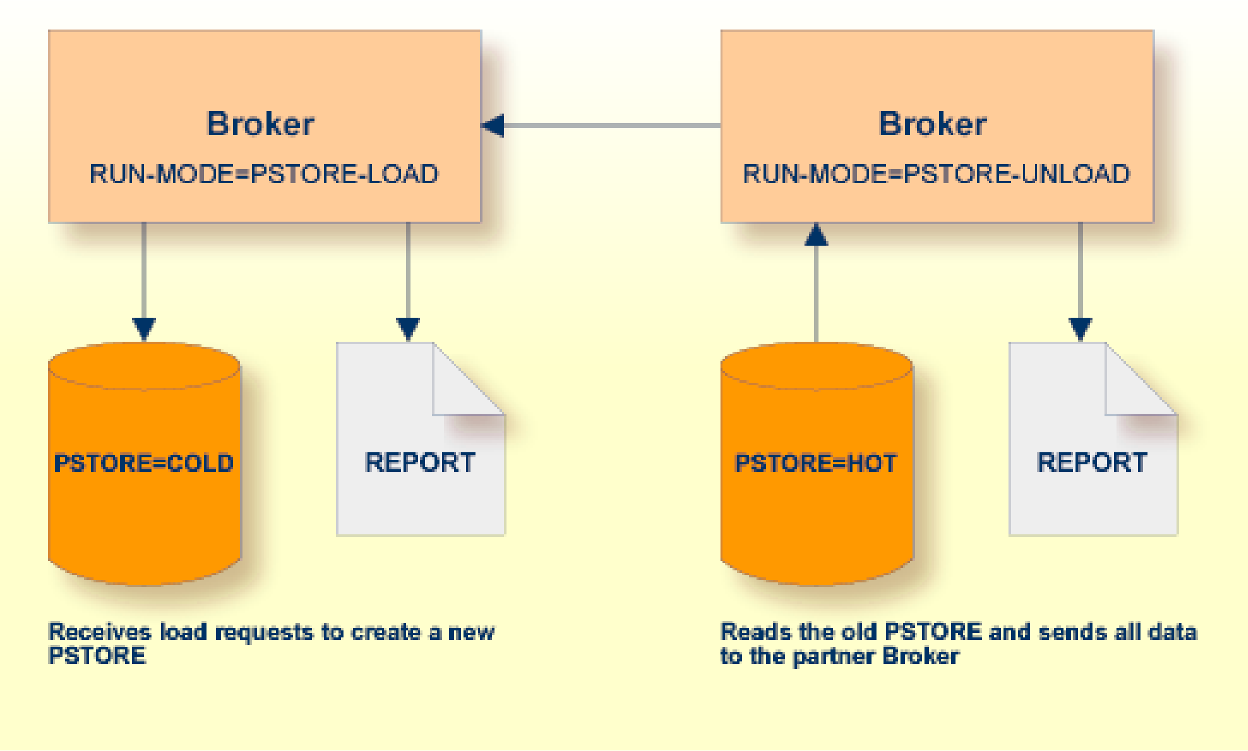
Migrating the Persistent Store

Introduction

The contents of EntireX Broker's persistent store can be migrated to a new persistent store in order to change the PSTORE type or to use the same type of PSTORE with increased capacity.

The migration procedure outlined here requires two Broker instances started with a special RUN-MODE parameter. One Broker unloads the contents of the persistent store and transmits the data to the other Broker, which loads data into the new PSTORE. Therefore, for the purposes of this discussion, we shall refer to an *unload* Broker and a *load* Broker.

This procedure is based on Broker-to-Broker communication to establish a communication link between two Broker instances. It does not use any conversion facilities, since the migration procedure is supported for homogeneous platforms only.


Broker
RUN-MODE=PSTORE-LOAD
Broker
RUN-MODE=PSTORE-UNLOAD
PSTORE=COLD
REPORT
PSTORE=HOT
REPORT
Receives load requests to create a new PSTORE
Reads the old PSTORE and sends all data to the partner Broker

Configuration

The migration procedure requires two Broker instances, each started with the RUN-MODE attribute. The unload Broker should be started with the following attribute:

```
RUN-MODE=PSTORE-UNLOAD
```

The load Broker should be started with the following attribute:

```
RUN-MODE=PSTORE-LOAD
```

These commands instruct the Broker instances to perform the PSTORE migration.

Note: The attribute PARTNER-CLUSTER-ADDRESS must be defined in both Broker instances to specify the transport address of the load Broker. The unload Broker must know the address of the load broker, and the load Broker must in turn know the address of the unload Broker.

Example:

Broker ETB001 performs the unload on host HOST1, and Broker ETB002 performs the load on host HOST2. The transmission is based on TCP/IP. Therefore, Broker ETB001 starts the TCP/IP communicator to establish port 1971, and Broker ETB002 starts the TCP/IP communicator to establish port 1972.

For ETB001, attribute PARTNER-CLUSTER-ADDRESS=HOST2:1972:TCP is set, and for ETB002, attribute PARTNER-CLUSTER-ADDRESS=HOST1:1971:TCP is set to establish the Broker-to-Broker communication between the two Broker instances.

In addition to attributes RUN-MODE and PARTNER-CLUSTER-ADDRESS, a fully functioning Broker configuration is required when starting the two Broker instances. To access an existing PSTORE on the unloader side, you must set the attribute PSTORE = HOT. To load the data into the new PSTORE on the loader side, you must set the attribute PSTORE=COLD. The load process requires an empty PSTORE at the beginning of the load process.

Note: Use caution not to assign PSTORE = COLD to your unload Broker instance, as this startup process will erase all data currently in the PSTORE.

For the migration process, the unload Broker and the load Broker must be assigned different persistent stores.

A report can be generated to detail all of the contents of the existing persistent store. At the end of the migration process, a second report can be run on the resulting new persistent store. These two reports can be compared to ensure that all contents were migrated properly. To run these reports, set the attribute PSTORE-REPORT = YES. See PSTORE under *Broker Attributes* for a detailed description, especially for the file assignment.

Migration Procedure

The migration procedure is made up of three steps.

Step 1

The unload Broker and the load Broker instances can be started independently of each other. Each instance will wait for the other to become available before starting the unload/load procedure.

The unload Broker instance sends a handshake request to the load Broker instance in order to perform an initial compatibility check. This validation is performed by Broker according to platform architecture type and Broker version number. The handshake ensures a correctly configured partner cluster address and ensures that the user did not assign the same PSTORE to both Broker instances. If a problem is detected, an error message will be issued and both Broker instances will stop.

Step 2

The unload Broker instance reads all PSTORE data in a special non-destructive raw mode and transmits the data to the load Broker instance. The load Broker instance writes the unchanged raw data to the new PSTORE. A report is created if PSTORE-REPORT=YES is specified, and a valid output file for the report is specified.

Step 3

The unload Broker instance requests a summary report from the load Broker instance to compare the amount of migrated data. The result of this check is reported by the unload Broker instance and the load Broker instance before they shut down.

When a Broker instances is started in `RUN-MODE=PSTORE-LOAD` or `RUN-MODE=PSTORE-UNLOAD`, the Broker instances only allow administration requests. All other user requests are prohibited.

Notes:

1. The contents of the persistent store are copied to the new persistent store as an exact replica. No filtering of unnecessary information will be performed - for example, UOWs in received state. The master records will not be updated.
2. Before restarting your Broker with the new persistent store, be sure to change your `PSTORE` attribute to `PSTORE=HOT`. *Do not* start your broker with the new persistent store using `PSTORE=COLD`; this startup process will erase all of the data in your persistent store.
3. After completing the migration process and restarting your Broker in a normal `RUN-MODE`, it is important not to bring both the new `PSTORE` and the old `PSTORE` back online using separate Broker instances; otherwise, applications would receive the same data twice. Once the migration process is completed satisfactorily, and is validated, the old `PSTORE` contents should be discarded.

Persistent Store Report

You can create an optional report file that provides details about all records added to or deleted from the persistent store. This section details how to create the report and provides a sample report.

- Configuration
- Sample Report

Configuration

To create a persistent store report, use Broker's global attribute `PSTORE-REPORT` with the value `YES`.

When the attribute value `YES` is supplied, all created or deleted persistent records will be reported if the output mechanism is available.

If the value `NO` is specified, no report will be created.

The report file is created using the following rules:

BS2000

`LINK-NAME ETBPREP` assigns the report file. Format `REC-FORM=V, REC-SIZE=0, FILE-TYPE ISAM` is used by default.

UNIX

Broker creates a file with the name *PSTORE.REPORT* in the current working directory. The file name *PSTORE.REPORT.LOAD* will be used if Broker is started with `RUN-MODE=PSTORE-LOAD`.

The file name *PSTORE.LOAD.UNLOAD* will be used if Broker is started with `RUN-MODE` = `PSTORE-UNLOAD`.

If the environment variable `ETB_PSTORE_REPORT` is supplied, the file name specified in the environment variable will be used.

If Broker receives the command-line argument `-p`, the token following argument `-p` will be used as the file name.

Windows

Same as UNIX.

z/OS

`DDNAME ETBPREP` assigns the report file. Format `RECFM=FB, LRECL=121` is used.

z/VSE

Logical unit `SYS003` and logical file name *ETBPREP* are used. Format `RECORD-FORMAT=FB`, `RECORD-LENGTH=121` is used.

Sample Report

The following is an excerpt from a sample `PSTORE` report.

```
EntireX 10.5     PSTORE Report      2016-10-18 10:46:18    Page     1

Identifier        Elements  Total length  Record Type   Date          Action
0000000000000000        1           760   Master        2016-10-18... Created
0010000000000001        1          5022   Conversation  2016-10-18... Created
0010000000000002        1          5022   Conversation  2016-10-18... Created
0010000000000003        1          5022   Conversation  2016-10-18... Created
0010000000000001                          Conversation  2016-10-18... Postponed
0010000000000001                          Conversation  2016-10-18... Accepted
0010000000000002                          Conversation  2016-10-18... Postponed
0010000000000002                          Conversation  2016-10-18... Accepted
0010000000000003                          Conversation  2016-10-18... Postponed
0010000000000003                          Conversation  2016-10-18... Accepted
0010000000000003                          Conversation  2016-10-18... Postponed
0010000000000003                          Conversation  2016-10-18... Accepted
0010000000000001                          Conversation  2016-10-18... Deleted
0010000000000002                          Conversation  2016-10-18... Deleted
0010000000000003                          Conversation  2016-10-18... Deleted
```

The following fields are provided in the report:

- `Identifier` provides the UOWID (record ID).
- `Elements` gives the number of messages per UOW when creating or loading records.
- `Total Length` gives the size of the raw record when creating or loading records.
- `Record Type` describes the type of the data. Following types are currently supported:
  - `Cluster`: a special record for synchronization purposes
  - `Conversation`: a unit of work as part of a conversation
  - `Master`: a special record to manage the persistent store
- Date and time of the action
- `Action` describes the action of Broker. The following actions are currently supported:
  - `Accepted`: UOW status was changed from `POSTPONED` to `ACCEPTED`
  - `Created`: record is created
  - `Deleted`: record is deleted
  - `Postponed`: UOW status was changed from `DELIVERED` to `POSTPONED`

  - `Loaded`: record is loaded (Broker instance with `RUN-MODE` = `PSTORE-LOAD`)
  - `Unloaded`: record is unloaded (Broker instance with `RUN-MODE` = `PSTORE-UNLOAD`)
- Remaining postpone attempts.

7 Using Persistence and Units of Work

This chapter describes implementation issues and how to use persistence and units of work in EntireX Broker. It assumes you are familiar with EntireX Broker from both an administrative and an application perspective, and with the ACI programming in particular. See also *EntireX Broker* and *EntireX Broker ACI Programming*.

Implementation Issues

Table of Persistent Store Drivers

A persistent store driver is an executable, or a load module that implements access to the physical persistent store. There is one persistent store driver for each persistent store type. The following table shows the persistent store options:

Persistent Store Type	Description	Operating System	Notes
Adabas	Uses Adabas database.	UNIX, Windows, z/OS, BS2000, z/VSE	Adabas, Software AG's ADAptable dataBASe, is a high-performance, multithreaded, database management system.
DIV	Uses IBM Data In Virtual facility on z/OS.	z/OS	This persistent store option is implemented as a VSAM linear data set.
CTREE	c-tree© is an embedded local database that can be used as your persistent store.	UNIX and Windows	c-tree© is the fast and reliable embedded database of FairCom Corporation®.

Changes are Required

It is important to note that some level of both application and system changes are necessary to utilize UOWs. Existing message-based Broker applications will continue to operate as before.

Attributes used for Units of Work

The following table represents the keyword parameters that can be used in the Broker attribute file for UOWs. A short form of the keyword is given if applicable. Default values are underlined.

Keyword	Value	Description
STORE	OFF | BROKER	Broker: sets default STORE attribute for all units of work. Service: sets default STORE attribute for units of work sent to the service.
MAX-UOWS or MUOW	0 | *n*	Broker: maximum number of active UOWs. If 0 is specified, then the Broker will not support any UOW operations. Service: maximum number of active UOWs for a service.
MAX-MESSAGES-IN-UOW or UMSG	16 | *n*	Broker: maximum number of messages in a UOW. Service: maximum number of messages in a UOW for the service.
PSTORE	NO | HOT | COLD | WARM	Broker only. Startup value for persistent store. NO No persistent store. HOT Persistent UOWs are restored to prior state during initialization. COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty. WARM (Internal Use Only) persistent UOWs are not restored during initialization, but the persistent store remains intact.
UWSTATP	0 - 254	Broker: persistent status is maintained either for persistent or non-persistent UOWs. Service: persistent status is maintained either for persistent or non-persistent UOWs for a service.
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	Broker: defines the lifetime of a UOW in seconds, minutes, hours or days. This value is the time that it can remain in the system without being completed. If the UOW is not completed within this time, it is deleted with a status of TIMEOUT Service: defines the lifetime of a UOW for a service.
MAX-UOW-MESSAGE-LENGTH	*n* | 31647	Broker: defines the default maximum message size that can be sent. Service: defines the maximum message size that can be sent to a service.

Keyword	Value	Description
DEFERRED	NO | YES	Broker: sets the default DEFERRED attribute for all services. UOWs can be sent to a deferred service even if the service is not registered. Service: sets the DEFERRED attribute for a service.

ACI Fields used for Units of Work

The following fields have been added to the broker ACI control block. Note that the actual field names may differ slightly depending on the programming language being used.

Keyword	Description
STORE	Indicates whether the specified UOW is persistent or not: OFF The sender accepts the persistence option specified by the service or Broker (this is the default value). BROKER The sender wants persistence. NO The sender does not want persistence, even if the service or Broker default is persistence. Also returned with RECEIVE to indicate if the UOW being received is persistent or not.
UWTIME	The amount of time that the UOW can remain incomplete without being timed out. This is also referred to as the UOW lifetime.
STATUS	The current status of a UOW. The status is returned on SEND, RECEIVE, and SYNCPOINT operations. Applicable values are as follows: RECEIVED One or more messages have been sent as part of a UOW but the UOW is not yet committed. ACCEPTED The UOW has been committed by the sender. DELIVERED The UOW is currently being received. POSTPONED The UOW was postponed by the receiver for later processing. BACKEDOUT * The UOW was backed out prior to being committed by the sender. PROCESSED * the receiver of the UOW has committed it. CANCELLED * the receiver of the UOW has cancelled it. TIMEOUT * the UOW was not processed within the specified time. DISCARDED * The UOW was not persistent and its data was discarded over a restart. * The status values marked with an asterisk are persistent, and will only exist for UOWs with persistent status. In addition, the following status values are returned on a RECEIVE operation to indicate if the message being received is part of a UOW or not, and if so, which part:

Keyword	Description
	RECV_NONE The message is not part of a UOW. RECV_FIRST The message is the first message in a UOW. RECV_MIDDLE The message is not the first or last message in a UOW. RECV_LAST The message is the last message in a UOW. RECV_ONLY The message is the only message in a UOW. All RECV_ values except RECV_NONE reflect an actual UOW status of DELIVERED.
USTATUS	A user-defined status associated with a UOW. It can be specified as part of a SEND, RECEIVE, or SYNCPOINT operation and will be returned whenever the status of a UOW is queried. See *Using User Status* below for more information.
UOWID	A unique identifier for a unit of work. This value is returned on SEND and RECEIVE operations and may be provided on SYNCPOINT operations that are querying status of UOWs.
UWSTATP	A numeric value indicating the lifetime value for persistent status. This value is a multiplier against the UWTIME value. Applicable values are: 0 Use the default specified for the service or broker. 1-254 Use 1 to 254 times the UWTIME value as the status lifetime. 255 The sender does not want persistent status, even if the service or broker default is persistent status.

ACI Function SYNCPOINT used for Units of Work

The SYNCPOINT function deals exclusively with UOWs. The following table lists the OPTION values that can be used with the SYNCPOINT function, and the associated behavior and restrictions of each one.

Note: In many cases, the behavior will be different depending on whether the issuer is the sender or the receiver of the UOW.

Option	Caller	Behavior and Restrictions
BACKOUT	Sender	If the specified UOW is in RECEIVED status, it will be put into BACKEDOUT status. If persistent status is not specified, no trace of the UOW will remain.
	Receiver	If the specified UOW is in DELIVERED status, it will be put back into ACCEPTED status and its attempted delivery count will be incremented.
CANCEL	Sender	If the specified UOW is in ACCEPTED status, it will be put into CANCELLED status. If persistent status is not specified, no trace of the UOW will remain.
	Receiver	If the specified UOW is in DELIVERED status, it will be put into CANCELLED status. If persistent status is not specified, no trace of the UOW will remain. If attributes POSTPONE-ATTEMPTS and POSTPONE-DELAY have been defined for the service, the UOW will be moved to the postpone queue instead of being deleted.

Option	Caller	Behavior and Restrictions
COMMIT	Sender	If the specified UOW is in RECEIVED status, it will be put into ACCEPTED status. It is now available to be received by the other partner.
	Receiver	If the specified UOW is in DELIVERED status, it will be put into PROCESSED status. If persistent status is not specified, no trace of the UOW will remain.
	Both	This is a special case of the COMMIT option, where the caller specifies UOWID=BOTH in the request. This allows the caller to commit two UOWs, one being received and one being sent, in a single atomic operation.
DELETE	Sender	Deletes the persistent status of the specified UOW. The UOW must be complete and must have been created by the caller. After this request, no trace of the UOW will remain.
EOC	Sender	Commits the UOW and sets an EOC indication on the associated conversation. See COMMIT for additional information and restrictions.
EOCCANCEL	Sender	Commits the UOW and sets an EOC-CANCEL indication on the associated conversation. See COMMIT for additional information and restrictions.
LAST	Sender	Returns the status of the last UOW sent by the caller. In addition, CLASS/SERVER/SERVICE details of the associated server are also returned. The CONV-ID can be included to qualify the request.
QUERY	Sender	With UOWID=n, returns the status of the specified UOW. In addition, CLASS/SERVER/SERVICE details of the associated server are also returned.
SETUSTATUS	Both	Updates the user status field of the specified UOW. The UOW must be in RECEIVED, ACCEPTED, or DELIVERED status.

Options used for UOW Operations

This table lists option values used to support UOW operations:

Option	Function	Behavior and Restrictions
SYNC	SEND	This option indicates that the data being sent is part of a UOW. The UOW is created on the first send, and subsequent sends will add messages to the UOW.
SYNC	RECEIVE	This option indicates that the RECEIVE can be satisfied only with a message in a UOW.
MSG	RECEIVE	This option indicates that the RECEIVE can be satisfied only with non-UOW messages.
ANY	RECEIVE	This option indicates that the RECEIVE can be satisfied by either a non-UOW or a UOW message. It is up to the receiver to determine which, based on the UOWSTATUS field that is returned.
COMMIT	SEND	This option combines a SEND and a SYNCPOINT,OPTION=COMMIT into a single operation. It allows the sender to create and commit a UOW in a single operation.

Using Units of Work

UOW vs non-UOW Conversations

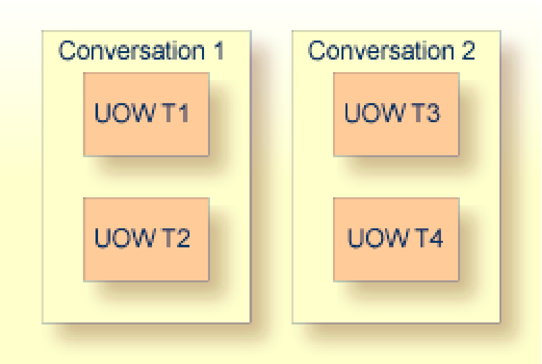
A Broker conversation will support either UOWs or messages, but not both. At the time the conversation is created, the Broker will determine which is to be supported.

Sequencing of Messages across Conversations

The order of delivery of new conversations to receivers is determined by the `COMMIT` time of the first UOW within its conversation. The conversation delivered to the receiver first is the one containing the first UOW for which the sender issues a `SEND,OPTION=COMMIT` or `SYNCPOINT,OPTION=COMMIT`.

If there is more than one UOW in a conversation, the `COMMIT` time of the first UOW determines the age of that conversation. Also, multiple UOWs within a conversation are picked up by the receiver, in the same sequence as they were committed by the sender.

Scenario: A server starts to receive UOWs (`CONVID=NEW`) and receives UOW T1 first, since this UOW is committed first. If the server issues another receive (`CONVID=NEW`), it receives UOW T3. If, however, the UOWs are not combined in conversations (i.e., every UOW is in a separate conversation), the server receives (`CONVID=NEW`) UOW T1 first, then UOW T2, UOW T3, etc.



The COMMITTIME field in the Broker control block shows COMMITTIME of the first UOW in a conversation.

Use of LOGON and TOKEN

An explicit LOGON function must be used before a program can use any of the UOW functions. In order to enable client and server programs to recover the status of their UOWs in the event of a failure (Broker, system, or application), these programs must specify a TOKEN value at the time of logon.

User Identification for Units of Work

EntireX Broker identifies participants by ACI fields USER-ID and TOKEN if TOKEN is supplied or by USER-ID and internal ID (so-called physical user ID) if TOKEN is not supplied. However, the implementation of persistent units of work is based on the user identification USER-ID and TOKEN.

Caution: In order to avoid unpredictable inconsistencies, all applications using persistent units of work must use this user identification to run correctly. The ACI verification routines do not restrict usage of UOWs to USER-ID and TOKEN yet. Modify your application accordingly.

Which Applications should use UOWs?

Applications that should consider using UOWs fit into a couple of different categories.

- Applications that currently use multiple messages to communicate a single request are good candidates for UOWs. Grouping these messages within a UOW can give the application additional control over how its data is processed.
- Applications that intend to utilize deferred services, persistence, or persistent status must use UOWs, since these facilities are not available to message-based applications.

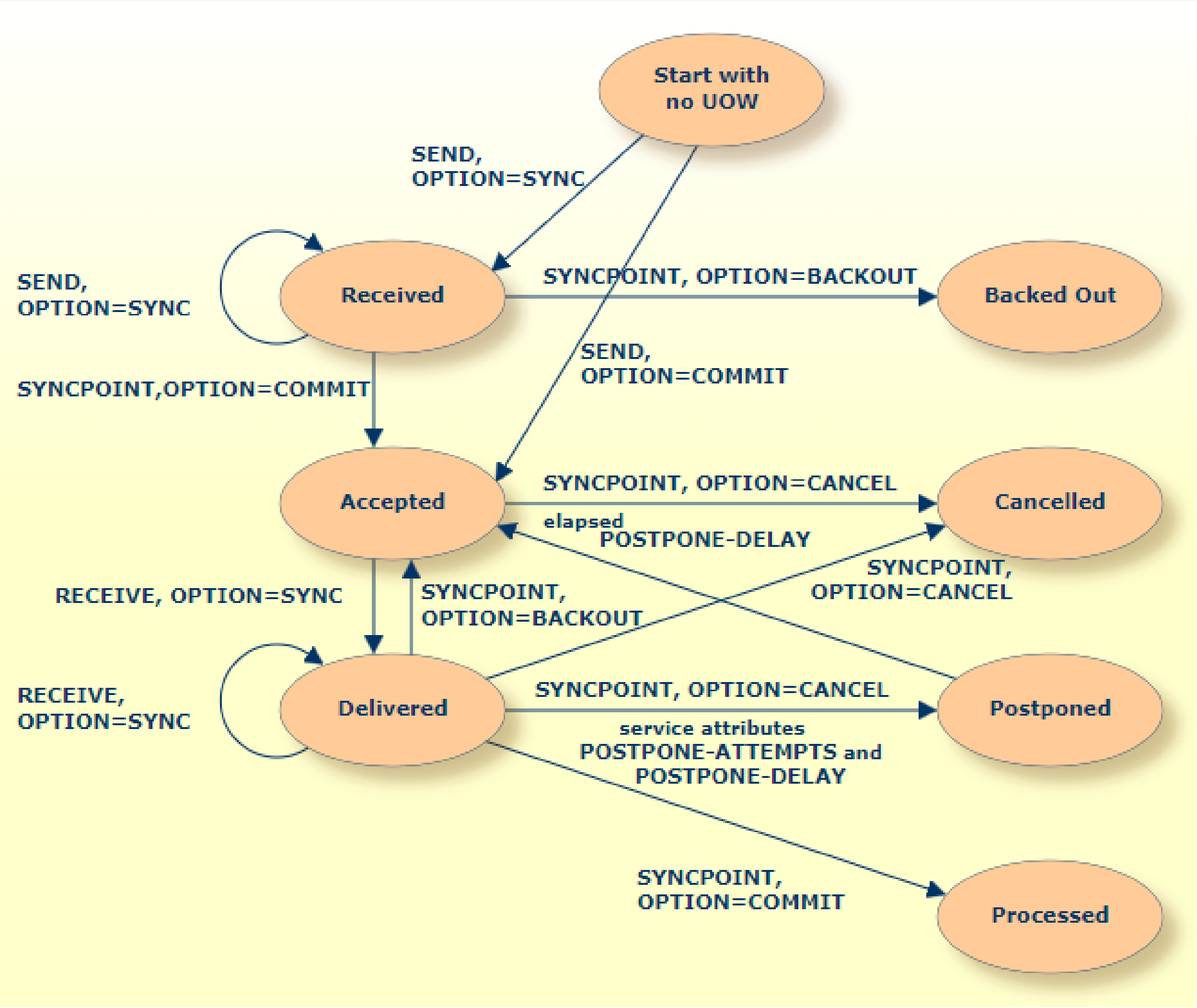
Understanding UOW Status

In order to use UOWs effectively, you need to understand

- the meaning of the various UOW status values;
- how they change based on events within the system;

  and
- how these changes are influenced by both persistence and persistent status.

The diagram below represents the normal status values as a UOW progresses through the system. These statuses and the transitions between them are not affected by either persistence or persistent status. The status values are indicated in ovals.



Normal Status Values as a UOW progresses through System

Note: The UOW is available to be received when it is first committed. The status values BACKEDOUT, CANCELLED and PROCESSED are valid only if there is persistent status.

UOW Status on RECEIVE

When a RECEIVE is issued for a message within a UOW, you might expect that the UOW status returned would be DELIVERED, since this is the actual status of the UOW. This is not the case, however. On a RECEIVE, the Broker returns a special UOW status that reflects additional information about the message and the UOW. These statuses are:

- RECV_FIRST= the message is the first message in a UOW.
- RECV_MIDDLE= the message is not the first or last message in a UOW.
- RECV_LAST= the message is the last message in a UOW.

- RECV_ONLY= the message is the only message in a UOW.
- RECV_NONE= the message is not part of a UOW. This status is particularly useful if the application is receiving both messages and UOWs.

If you receive a status of either RECV_LAST or RECV_ONLY and then issue another RECEIVE for the same UOW, you will get an error 00740301 Conversation found: end of unit of work indicating the end of the UOW.

Using User Status

The user status field of the UOW allows additional, application-specific information to be carried with the UOW. It can be used to maintain status or indicate error information. It can also provide a form of "out-of-band" data communication between the sender and the receiver of a UOW.

For example, if a server is processing a long-running UOW, it can periodically update the user status of the UOW (using SYNCPOINT, OPTION=SETUSTATUS) to indicate its progress. The client can periodically get the user status (using SYNCPOINT, OPTION=QUERY) and report the progress back to the end-user.

As another example, the sender of a long-running UOW can update the user status to indicate that processing of the UOW should be abandoned by the server. The server can periodically get the user status while processing and react accordingly.

Resource and Performance Considerations

Each active UOW consumes memory resources (approximately 140 bytes per UOW) in a preallocated pool, not including the size of the message itself.

Also, additional memory resources such as the conversation and participant control blocks for the UOW, together with messages associated with them, will remain in memory for a deferred service when persistence is used. This can become significant when UOWs are being sent to a deferred service. However, the message itself does not remain in memory if sent to a service which is not currently registered - the whole purpose of deferred services. If the service is currently registered, the message remains in memory.

Messages that are sent to any (registered or unregistered) service can be "paged out" by Broker if storage is required. This feature considerably eases memory consumption when using persistence.

Using Persistence

When do Persistent UOWs make Sense?

A UOW should be made persistent if the sender wants the Broker to assure that the UOW will be deliverable, even if there is a system or Broker failure. Assured delivery assumes that the intended receiver of the UOW is active, or becomes active within the specified lifetime of the UOW.

Since most existing Broker applications are interactive, they are probably not good candidates for persistent UOWs. New application models can now be implemented, using persistent UOWs. For example, a service that collects information from other services, such as accounting, inventory, logging, etc., would be a good fit for persistent UOWs. Another example could be a client sending a long-running request to a service (one that may be inactive or busy), disconnecting, and coming back some time later to retrieve the results. The reliability of assured delivery makes this model practical.

Persistent UOWs do not require persistent status.

Adding Persistence to a UOW

A UOW can be made persistent:

- by specifying `STORE=BROKER` in the ACI request that creates the UOW;
- by specifying `STORE=BROKER` in service definition or service defaults portion of the Broker attribute file, making all UOWs for that service persistent; or
- by specifying `STORE=BROKER` in the Broker defaults section of the Broker attribute files, making all UOWs in the system persistent.

In addition, specifying `STORE=NO` in the ACI request that creates the UOW will explicitly make the UOW non-persistent, overriding any Broker or service default.

Resource and Performance Considerations

A persistent UOW consumes resources in two areas.

- When the UOW is committed by the sender, all of the messages are written to the persistent store. This will generate multiple I/O operations, depending on the number and size of the messages.
- Space used to store the UOW and its messages will be allocated in the persistent store and will remain until the UOW is completed.

Performance of certain specific functions (e.g. `SYNCPOINT OPTION=COMMIT` by the sender of a UOW) will be affected by the additional time required to perform the I/O operations associated with writing the UOW and message(s) to the persistent store. These operations are performed synchronously because the Broker must ensure that the UOW, once committed, can be recovered in the event of a system or Broker failure.

Which Information is saved with the UOW?

When the UOW is initially created in the persistent store, the following information is written:

- Unit-of-work ID
- Conversation ID
- UOW Sender information, including:
  - User ID
  - Token
  - Server/service/class *
- UOW receiver information, including:
  - User ID **
  - Token **
  - Server/service/class *
- Creation timestamp
- UOW lifetime value
- Persistence and persistent status values

The following pieces of information will be included when the UOW is initially written to the persistent store and will be updated, as needed, during the life of the UOW:

- UOW status
- UOW user status
- Attempted delivery count

- Number of messages in UOW
- Total message size in UOW
- Persistent status lifetime value
- Conversation state and EOC reason code

* Server/service/class information is only saved if the sender or receiver is a registered service.

** The receiver's user ID and token are only saved if the receiver is a service that has already acquired the conversation associated with this UOW. When there are multiple instances of a service, this means that a new conversation can be restarted by any instance of the service, but an existing conversation is bound to a specific instance of the service.

What happens when Broker restarts?

- Restart Behavior of UOW
- Re-creation of Internal Control Blocks
- Behavior of Conversation at Broker Restart

Note: "Restored" is an active UOW which has been returned to `ACCEPTED` status; "Discarded" is a UOW which has not been returned to `ACCEPTED` status. "Discarded" does not imply the status of `DISCARDED`.

Caution: The persistent store must be available before you attempt to restart your Broker; otherwise your Broker will not restart.

Restart Behavior of UOW

- **Restart Table 1**
  The behavior during restart of the following states depends on the previous settings of the options Persistent UOW and Persistent Status.

UOW Status before Restart	Persistent UOW: YES | NO	Persistent Status: YES | NO	Behavior of UOW and Status	UOW Status after Restart *
RECEIVED	YES	YES	UOW not restored; Status is restored	BACKEDOUT
RECEIVED	YES	NO	UOW not restored; Status not restored	---
RECEIVED	NO	YES	UOW not restored; Status is restored	DISCARDED
RECEIVED	NO	NO	UOW not restored; Status not restored	---
ACCEPTED	YES	YES	UOW is restored; Status is restored	ACCEPTED

UOW Status before Restart	Persistent UOW: YES | NO	Persistent Status: YES | NO	Behavior of UOW and Status	UOW Status after Restart *
ACCEPTED	YES	NO	UOW is restored; Status is restored	ACCEPTED
ACCEPTED	NO	YES	UOW not restored; Status is restored	DISCARDED
ACCEPTED	NO	NO	UOW not restored; Status not restored	---
DELIVERED	YES	YES	UOW is restored; Status is restored	ACCEPTED
DELIVERED	YES	NO	UOW is restored; Status is restored	ACCEPTED
DELIVERED	NO	YES	UOW not restored; Status is restored	DISCARDED
DELIVERED	NO	NO	UOW not restored; Status not restored	---
POSTPONED	YES	YES	UOW is restored; Status is restored	ACCEPTED
POSTPONED	YES	NO	UOW is restored; Status is restored	ACCEPTED
POSTPONED	NO	YES	UOW is not restored; Status is restored	DISCARDED
POSTPONED	NO	NO	UOW is not restored; Status is not restored	---
PROCESSED **	YES	YES	Status is restored	PROCESSED
PROCESSED **	YES	NO	Status is not restored	---
PROCESSED **	NO	YES	Status is restored	PROCESSED
PROCESSED **	NO	NO	Status not restored	---

* If either UOW or its status is restored.

** In this state, the UOW information has already been deleted upon reaching PROCESSED status.

- **Restart Table 2**
  The behavior during restart of the following states does not depend on the settings of Persistent UOW; in these cases only the Persistent Status exists and does not change after a restart. There is no UOW to be restored.

UOW Status before Restart	Behavior of Status	UOW Status after Restart
CANCELLED	Status is restored	CANCELLED
DISCARDED	Status is restored	DISCARDED
BACKEDOUT	Status is restored	BACKEDOUT
TIMEDOUT	Status is restored	TIMEDOUT

Re-creation of Internal Control Blocks

To restore a UOW, the Broker re-creates all internal control blocks necessary to represent the UOW when it was accepted. The table displays the targets of each control block type:

Control Block Type	Association: Sender | Receiver	Notes
PCB	Sender; Receiver (optional)	PCB = Participant CB
SCB	Sender; Receiver	SCB = Service CB
CCB	Sender; Receiver	CCB = Conversation CB Two CCBs represent the conversation.
UOW	Receiver	UOW = unit of work CB

Note: The messages associated with the UOW are not re-created in memory until a RECEIVE is actually issued for the UOW.

Behavior of Conversation at Broker Restart

Broker sets any units of work (UOWs) that are in DELIVERED status to ACCEPTED status during restart processing. If this is the first unit of work within a conversation sent by a client to a server, the assignment of the conversation to a particular server is dropped and the conversation is again available for all servers offering the same service.

If there is more than one unit of work in a single conversation and the first UOW is already received and committed by the server, the link to the server will kept even after this (non-first) UOW has reverted from DELIVERED to ACCEPTED status during restart processing. The server can retrieve units of work after restart with function RECEIVE OPTION=SYNC,CONVID=ANY and will get all old conversations containing UOWs first and then new conversations containing UOWs.

Servers performing a RECEIVE OPTION=SYNC, CONVID=NEW will retrieve only conversations not already assigned to this server. We strongly recommend that you implement RECEIVE OPTION=SYNC,CONVID=ANY or CONVID=OLD to retrieve already assigned conversations.

UOWs and Replicated Servers

Special consideration must be given when restarts occur, and there are persistent UOWs that are being sent to replicated servers, e.g. when more than one copy of a server is active. This is because a UOW is not associated with a server instance until the UOW's conversation is actually received by a server. From an application perspective, this means that a conversation that has not yet been received by its target server will be restored so that any instance of the server can process it. However, once the conversation has been received, any subsequent UOWs sent on the conversation will be restored so that only the specific instance, based on USER-ID and TOKEN, can receive them. The reasoning behind this is that a broker restart can occur without the servers being restarted, and the servers could be maintaining context information based on the conversation.

It is important to note that this can cause problems if the server instances are started as a result of load and the same load conditions are not present after the restart. For example, a UOW could be bound to the fifth instance of a server, but after a restart there is only enough load to start three instances. For this reason, we recommend that replicated servers using persistent UOWs not maintain any conversations with multiple UOWs.

Postponing Units of Work

A received unit of work has to be committed to indicate successful completion. However, if processing of the UOW is temporarily not possible, the receiver issues a SYNCPOINT,OPTION=BACKOUT function to set it to ACCEPTED state again, or issues SYNCPOINT,OPTION=CANCEL to delete the UOW. The receiver will get the UOW again due to BACKOUT, or the UOW is deleted due to CANCEL.

If such a temporary outage occurs for certain services, you can configure a postponement of units of work in the Broker attribute file. Define your postpone queue with service-specific attributes POSTPONE-ATTEMPTS and POSTPONE-DELAY. The receiver still issues SYNCPOINT,OPTION=CANCEL. In this case, CANCEL moves the UOW to the postpone queue instead of deleting it. The UOW gets status POSTPONED and is no longer accessible until the time defined with POSTPONE-DELAY has elapsed.

When the POSTPONE-DELAY has elapsed, the UOW gets status ACCEPTED again and is moved back to the queue of available UOWs. The receiver can now process the UOW, but if the outage or the lack of resources could not be fixed in the meantime, the UOW can be postponed again with SYNCPOINT,OPTION=CANCEL. The value for attribute POSTPONE-ATTEMPTS defines the maximum number of possible postpone attempts.

This postpone handling will not change the lifetime of the UOW, which means that the POSTPONE-DELAY multiplied by the number of POSTPONE-ATTEMPTS should be lower than UOW-DATA-LIFETIME.

The sequence of UOWs (commit time of the producer in ascending order) cannot be guaranteed when UOWs have been postponed and brought back to ACCEPTED state. This applies also to operations with ETBCMD to modify the status of UOWs. See SET-UOW-STATUS in command-line utility ETBCMD (z/OS | UNIX | Windows).

Using Persistent Status

When does Persistent Status make Sense?

Persistent status should be considered for applications in which the sender needs to know if UOWs were actually processed successfully. In cases where the data associated with a UOW can be easily re-created in the event of a failure, persistent status may be a more desirable and lower-overhead alternative to a persistent UOW.

Persistent status does not require a persistent UOW.

Adding Persistent Status to a UOW

A UOW's status can be made persistent:

- by specifying a `UWSTATP` value between 1 and 254 in the ACI request that creates the UOW;
- by specifying a `UWSTATP` value between 1 and 254 in service definition or service defaults portion of the Broker attribute file, making the status of all UOWs for that service persistent; or
- by specifying a `UWSTATP` value between 1 and 254 in the Broker defaults section of the Broker attribute files, making the status of all UOWs in the system persistent.

Specifying `UWSTATP=255` in the ACI request that creates the UOW will explicitly make the UOW status non-persistent, overriding any broker or service default.

Resource and Performance Considerations

Using persistent status consumes resources in two areas.

- The persistent store is updated each time the UOW is modified, by either the sender or the receiver. These modifications occur whenever a `SEND` or `RECEIVE` function is issued for the UOW, or whenever its status is changed, such as by `SYNCPOINT OPTION=COMMIT`. Depending on the implementation, this will generate one or more I/O operations.
- The space used for the UOW (but not its messages) in the persistent store remains allocated for some period of time after the UOW has been completed.

The performance of individual requests will generally be affected by the additional time required to perform the I/O operations associated with maintaining persistent status. At this time, all operations are performed synchronously, although that may change in future releases.

Recovery Processing

Introduction

UOWs and persistence provide functionality for the application program (either client or server) to recover from failures: i.e., system, broker or application. In addition, this functionality allow new types of applications to be built, including ones not requiring concurrent execution of the client and server.

There are no standard rules for recovery, because each application model will use this functionality differently and will have different requirements for recovery. But the considerations in the following section should be kept in mind.

Determining the Status of a UOW

The most useful function for recovery is the `SYNCPOINT`, `OPTION=LAST`. This function will return the UOWID, CID, and status of the last UOW created by the caller, based on the `USER-ID` and `TOKEN`. This function can be used when an application starts or when it detects a failure to determine how much processing has been completed on a UOW. This information can then be used to decide how to recover from the failure.

A Real-world Example: Chess-by-Mail

Chess-by-mail is a sample of an application that takes advantage of UOWs, persistence, and persistent status. In generic terms, this application involves a client and a server exchanging messages on a single conversation. The conversation is long-running, and there is no requirement that the client and the server be active at the same time.

Although chess-by-mail was conceived as a single application, it is perhaps easier to describe its operation separately for the client and the server side. By convention, the white player is the client and the black player is the server. For simplicity, any user interaction has been left out of the description. Also for simplicity, only one chess-by-mail game is assumed to be running at any one time.

- Client Behavior
- Server Behavior

Client Behavior

The behavior of the chess-by-mail client is as follows:

1. Logon, specifying a `USER-ID` and `TOKEN`, which allow recovery of prior UOWs.
2. Issue `SYNCPOINT`, `OPTION=LAST` to determine the status of the last UOW created.
3. If the return code is `00780305 - UOW not found`, then there is no game in progress. So send the first white move to the server with: `SEND OPTION=COMMIT,CID=NEW`. If the send is successful, logoff and exit.
4. If the return code from `SYNCPOINT` is 0, then there is a last UOW and therefore a game is in progress. The UOW status value is examined to decide how to proceed.
5. If the status is `ACCEPTED`, then the server has not yet received the last move, so logoff and exit.
6. If the status is `DELIVERED`, then the server is currently processing the last move, so logoff and exit.
7. If the status is `TIMEOUT`, then the server did not receive the last move before its lifetime expired, so logoff and exit.
8. If the status is `PROCESSED`, then the server has received the last move and committed the UOW. Our application model has the client sending a move in response and committing both UOWs at the same time. So we need to receive the new move and send a reply to it.
9. Get the server's move with `RECEIVE,OPTION=SYNC,CID=`*n*, where *n* is the CID returned from `SYNCPOINT OPTION=LAST`.
10. Send the response move back using `SEND OPTION=SYNC,CID=n`.
11. Commit both the received and sent UOWs with a single call `SYNCPOINT OPTION=COMMIT,UOWID=BOTH`.
12. Logoff and exit.

Server Behavior

The behavior of the chess-by-mail server is as follows:

1. Logon, specifying a Userid and Token, which allow recovery of prior UOWs.
2. Register as the chess-by-mail server.
3. Issue `SYNCPOINT OPTION=LAST` to determine the status of the last UOW created.
4. If the return code is `00780305 - UOW not found`, then there is no game in progress. So we receive first white move from the client with: `RECEIVE OPTION=SYNC,CID=NEW`. When the `RECEIVE` has been completed, continue at step 11.
5. If the return code from `SYNCPOINT` is 0, then there is a last UOW and therefore a game is in progress. The UOW status value is examined to decide how to proceed.
6. If the status is `ACCEPTED`, then the client has not yet received the last move, so deregister, logoff and exit.
7. If the status is `DELIVERED`, then the client is currently processing the last move, so deregister, logoff and exit.
8. If the status is `TIMEOUT`, then the client did not receive the last move before its lifetime expired, so deregister, logoff and exit.
9. If the status is `PROCESSED`, then the client has received the last move and committed the UOW. Our application model has the server sending a move in response and committing both UOWs at the same time. So we need to receive the new move and send a reply to it.
10. Get the client's move with `RECEIVE,OPTION=SYNC,CID=`*n*, where *n* is the CID returned from

    `SYNCPOINT,OPTION=LAST`.
11. Send the response move back using `SEND,OPTION=SYNC,CID=`*n*.
12. Commit both the received and sent UOWs with a single call:

    `SYNCPOINT,OPTION=COMMIT,UOWID=BOTH`.
13. Deregister, logoff and exit.

8 Broker UOW Status Transition

This chapter contains the UOW status transition tables for EntireX Broker and covers the following topics:

See also *Broker ACI Fields* | *Broker ACI Functions* | *Error Messages and Codes*.

Initial UOW Status: NULL | Received

Initial UOW Status	Action	Resulting UOW Status				Description
		PU&PS	PU&NPS	NPU&PS	NPU&NPS	
Received	Send	Received	Received	Received	Received	
Received	Commit	Accepted	Accepted	Accepted	Accepted	
Received	ReStart	BackedOut	NULL	Discarded	NULL	
Received	BackOut	BackedOut	NULL	BackedOut	NULL	
Received	TimeOut	BackedOut	NULL	BackedOut	NULL	R6: This action can only be a conversation timeout since a UOW only exists once it is committed.
Received	Delete	Received	Received	Received	Received	
Received	Cancel	Received	Received	Received	Received	
Received	Receive	Received	Received	Received	Received	

Initial UOW Status: Accepted | Delivered | Postponed

Initial UOW Status	Action	Resulting UOW Status				Description
		PU&PS	PU&NPS	NPU&PS	NPU&NPS	
Accepted	Receive	Delivered	Delivered	Delivered	Delivered	
Accepted	Timeout	Timedout	NULL	Timedout	NULL	
Accepted	Restart	Accepted	Accepted	Discarded	NULL	
Accepted	Cancel	Cancelled	NULL	Cancelled	NULL	
Accepted	Delete	Accepted	Accepted	Accepted	Accepted	
Accepted	BackOut	Accepted	Accepted	Accepted	Accepted	
Accepted	Send	Accepted	Accepted	Accepted	Accepted	
Accepted	Commit	Accepted	Accepted	Accepted	Accepted	
Delivered	Receive	Delivered	Delivered	Delivered	Delivered	
Delivered	Commit	Processed	NULL	Processed	NULL	
Delivered	Cancel	Cancelled	NULL	Cancelled	NULL	R20: Cancel can only be issued by receiver of the UOW.
Delivered	BackOut	Accepted	Accepted	Accepted	Accepted	
Delivered	TimeOut	Timedout	NULL	NULL	NULL	
Delivered	Restart	Accepted	Accepted	Discarded	NULL	
Delivered	Delete	Delivered	Delivered	Delivered	Delivered	
Delivered	Send	Delivered	Delivered	Delivered	Delivered	
Postponed	Receive	N/A	N/A	N/A	N/A	Receive cannot be issued by any user
Postponed	Commit	N/A	N/A	N/A	N/A	Commit cannot be issued by any user.
Postponed	Cancel	Cancelled	NULL	Cancelled	NULL	Cancel can only be issued by the sender of the UOW.
Postponed	BackOut	N/A	N/A	N/A	N/A	BackOut cannot be issued by any user.
Postponed	TimeOut	Timedout	NULL	NULL	NULL	
Postponed	Restart	Accepted	Accepted	Discarded	NULL	
Postponed	Delete	N/A	N/A	N/A	N/A	Delete cannot be issued by any user.
Postponed	Send	N/A	N/A	N/A	N/A	Send cannot be issued by any user.

Initial UOW Status: Processed | Timedout

Initial UOW Status	Action	Resulting UOW Status				Description
		PU&PS	PU&NPS	NPU&PS	NPU&NPS	
Processed	Delete	NULL	N/A	NULL	N/A	Processed is a STABLE UOW status:
Processed	Timeout	NULL	NULL	NULL	N/A	All actions and transitions refer to the status of a UOW.
Processed	Restart	Processed	N/A	Processed	N/A	
Processed	Backout	Processed	N/A	Processed	N/A	
Processed	Cancel	Processed	N/A	Processed	N/A	
Processed	Commit	Processed	N/A	Processed	N/A	
Processed	Receive	Processed	N/A	Processed	N/A	
Processed	Send	Processed	N/A	Processed	N/A	
Timedout	Restart	Timeout	N/A	Timeout	N/A	Timedout is a STABLE UOW status:
Timedout	Delete	NULL	N/A	NULL	N/A	All actions and transitions refer to the status of a UOW.
Timedout	Timeout	NULL	N/A	NULL	N/A	
Timedout	Send	Timedout	N/A	Timedout	N/A	
Timedout	Receive	Timedout	N/A	Timedout	N/A	
Timedout	Commit	Timedout	N/A	Timedout	N/A	
Timedout	Backout	Timedout	N/A	Timedout	N/A	
Timedout	Cancel	Timedout	N/A	Timedout	N/A	

Initial UOW Status: Cancelled | Discarded | Backedout

Initial UOW Status	Action	Resulting UOW Status				Description
		PU&PS	PU&NPS	NPU&PS	NPU&NPS	
Cancelled	Delete	NULL	N/A	NULL	N/A	Cancelled is a STABLE UOW status:
Cancelled	Restart	Cancelled	N/A	Cancelled	N/A	All actions and transitions refer to the status of a UOW.
Cancelled	TimeOut	NULL	N/A	NULL	N/A	
Cancelled	Send	Cancelled	N/A	Cancelled	N/A	
Cancelled	Receive	Cancelled	N/A	Cancelled	N/A	
Cancelled	Commit	Cancelled	N/A	Cancelled	N/A	
Cancelled	Backout	Cancelled	N/A	Cancelled	N/A	
Cancelled	Cancel	Cancelled	N/A	Cancelled	N/A	
Discarded	Delete	N/A	N/A	NULL	N/A	Discarded is a STABLE UOW status:
Discarded	TimeOut	N/A	N/A	NULL	N/A	All actions and transitions refer to the status of a UOW.
Discarded	Restart	N/A	N/A	Discarded	N/A	
Discarded	Cancel	N/A	N/A	Discarded	N/A	
Discarded	Send	N/A	N/A	Discarded	N/A	
Discarded	Receive	N/A	N/A	Discarded	N/A	
Discarded	Commit	N/A	N/A	Discarded	N/A	
Discarded	Backout	N/A	N/A	Discarded	N/A	
BackedOut	TimeOut	NULL	N/A	NULL	N/A	BackedOut is a STABLE UOW status:
BackedOut	Cancel	BackedOut	N/A	BackedOut	N/A	All actions and transitions refer to the status of a UOW
BackedOut	Restart	BackedOut	N/A	BackedOut	N/A	
BackedOut	Send	BackedOut	N/A	BackedOut	N/A	
BackedOut	Receive	BackedOut	N/A	BackedOut	N/A	
BackedOut	Commit	BackedOut	N/A	BackedOut	N/A	
BackedOut	Delete	NULL	N/A	NULL	N/A	
BackedOut	Backout	BackedOut	N/A	BackedOut	N/A	

Legend for UOW Status Transition Table

Abbreviation	Resulting UOW Status
N/A	Not applicable
UOW Status	Error condition, message issued, no change

Table of Column Abbreviations

Abbreviation	UOW Status
PU	Persistent unit of work
PS	Persistent status
NPU	Non-persistent unit of work
NPS	Non-persistent status

9 Accounting in EntireX Broker

This chapter describes the accounting records for Broker that can be used for several purposes, including:

- **application chargeback**
  for apportioning EntireX resource consumption on the conversation and/or the application level;
- **performance measurement**
  for analyzing application throughput (bytes, messages, etc.) to determine overall performance;
- **trend analysis**
  for using data to determine periods of heavy and/or light resource and/or application usage.

EntireX Accounting Data Fields

In the EntireX Accounting record, there are various types of data available for consumption by applications that process the accounting data:

Field Name	Accounting Version	Type of Field	Description
SMF Record Type	1	1-byte unsigned integer	z/OS only. Type of SMF record.
Record Write Time	1	z/OS: I4I4 timestamp Other platforms: A14 timestamp	z/OS: SMF timestamp in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)). Other platforms: The time this record was written to the accounting file in "YYYYMMDDHHMMSS" format.
SMF system ID	1	4-byte string	z/OS only. ID of the SMF system.
SMF subsystem ID	1	4-byte string	z/OS only. ID of the SMF subsystem.
EntireX Broker ID	1	A32	Broker ID from attribute file.
EntireX Version	1	A8	Version information, *v.r.s.p* where *v* =version *r* =release *s* =service pack *p* =patch level for example 10.5.0.00.
Platform of Operation	1	A32 (A8 under z/OS)	Platform where EntireX is running.

Field Name	Accounting Version	Type of Field	Description
EntireX Start Time	1	z/OS: I4I4 timestamp Other platforms: A14 timestamp	z/OS: The time EntireX was initialized in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)). Other platforms: The time EntireX was initialized in "YYYYMMDDHHMMSS" format.
Accounting Record Type	1	A1	It is always C for conversation. Future Types will have a different value in this field.
Client User ID	1	A32	`USER-ID` ACI field from the client in the conversation.
Client Token	1	A32	`TOKEN` field from the ACI from the client.
Client Physical ID	1	A56	The physical user ID of the client, set by EntireX.
Client Communication Type	1	I1	Communication used by client: 1 = Net-Work 2 = TCP/IP 3 = APPC 4 = IBM® MQ 5 = SSL
Client Requests Made	1	I4	Number of Requests made by client.
Client Sent Bytes	1	I4	Number of bytes sent by client.
Client Received Bytes	1	I4	Number of bytes received by client.
Client Sent Messages	1	I4	Number of messages sent by client.
Client Received Messages	1	I4	Number of messages received by client.
Client Sent UOWs	1	I4	Number of UOWs sent by client.
Client UOWs Received	1	I4	Number of UOWs received by client.
Client Completion Code	1	I4	Completion code client received when conversation ended.
Server User ID	1	A32	`USER-ID` ACI field from the server in the conversation.
Server Token	1	A32	`TOKEN` field from the ACI from the server.
Server Physical ID	1	A56	The physical user ID of the server, set by EntireX.
Server Communication Type	1	I1	Communication used by Server: 1 = Entire Net-Work 2 = TCP/IP 3 = APPC 4 = IBM® MQ 5 = SSL
Server Requests Made	1	I4	Number of requests made by server.
Server Sent Bytes	1	I4	Number of bytes sent by server.

Field Name	Accounting Version	Type of Field	Description
Server Received Bytes	1	I4	Number of bytes received by server.
Server Sent Messages	1	I4	Number of messages sent by server.
Server Received Messages	1	I4	Number of messages received by server.
Server Sent UOWs	1	I4	Number of UOWs sent by server.
Server Received UOWs	1	I4	Number of UOWs received by server.
Server Completion Code	1	I4	Completion code server received when conversation ended.
Conversation ID	1	A16	`CONV-ID` from ACI.
Server Class	1	A32	`SERVER-CLASS` from ACI.
Server Name	1	A32	`SERVER-NAME` from ACI.
Service Name	1	A32	`SERVICE` from ACI.
CID=NONE Indicator	1	A1	Will be N if `CONV-ID=NONE` is indicated in application.
Restarted Indicator	1	A1	Will be R if a conversation was restarted after a Broker shutdown.
Conversation Start Time	1	z/OS: I4I4 timestamp Other platforms: A14 timestamp	z/OS: The time the conversation began in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)). Other platforms: The time the conversation began in "YYYYMMDDHHMMSS" format.
Conversation End Time	1	z/OS: I4I4 timestamp Other platforms: A14 timestamp	z/OS: The time the conversation was cleaned up in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)). Other platforms: The time the conversation was cleaned up in "YYYYMMDDHHMMSS" format.
Conversation CPU Time	1	I4	Number of microseconds of CPU time used by the conversation
Client Security Identity	2	A32	Actual identity of client derived from authenticated user ID.
Client Application Node	2	A32	Node name of machine where client application executes.
Client Application Type	2	A8	Stub type used by client application.
Client Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field `APPLICATION-NAME`.
Client Credentials Type	2	I1	Mechanism by which authentication is performed for client.

Field Name	Accounting Version	Type of Field	Description
Server Security Identity	2	A32	Actual identity of server derived from authenticated user ID.
Server Application Node	2	A32	Node name of machine where server application executes.
Server Application Type	2	A8	Stub type used by server application.
Server Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field `APPLICATION-NAME`.
Server Credentials Type	2	I1	Mechanism by which authentication is performed for server.
Client RPC Library	3	A128	RPC library referenced by client when sending the only/first request message of the conversation.
Client RPC Program	3	A128	RPC Program referenced by client when sending the only/first request message of the conversation.
Server RPC Library	3	A128	RPC library referenced by server when sending the only/first response message of the conversation.
Server RPC Program	3	A128	RPC Program referenced by server when sending the only/first response message of the conversation.
Client IPv4 Address	4	A16	IPv4 address of the client.
Server IPv4 Address	4	A16	IPv4 address of the server.
Client Application Version	4	A16	Application version of the client.
Server Application Version	4	A16	Application version of the server.
Client IPv6 Address	5	A46	IPv6 address of the client.
Server IPv6 Address	5	A46	IPv6 address of the server.

Note: Accounting fields of any version greater than 1 are created only if the attribute `ACCOUNTING-VERSION` value is greater than or equal to the corresponding version. For example: accounting fields of version 2 are visible only if `ACCOUNTING-VERSION=2` or higher is specified.

Using Accounting under UNIX and Windows

Broker Attribute File Settings

ACCOUNTING = NO | YES | (YES, SEPARATOR=Separator Characters) (Default is NO)

Set this parameter to "NO" (i.e., do not create accounting data) or "YES" to create accounting data. Up to seven separator characters can specified using the SEPARATOR suboption, for example ACCOUNTING = (YES, SEPARATOR=;). If no separator character is specified, the comma character will be used.

Retrieving Accounting Data

The accounting file will be located in the Broker's installed directory. The file's name is based on the ETB_LOG environment variable and the current date and time (for uniqueness). Example: If ETB_LOG is set to BROKER1.LOG, the accounting data file will be named BROKER1_*YYYYMMDDHHMMSS*.csv. If ETB_LOG is not set, the Broker's ID will be used, with an extension of CSV (e.g. ETB048_*YYYYMMDDHHMMSS*.csv). See *Environment Variables in EntireX*.

Using Accounting under z/OS

The `ACCOUNTING` attribute indicates if accounting records will be generated. Accounting records are written upon successful completion of a conversation. A conversation ending in an application error (such as a timeout) is considered to be a successful conversation.

Attribute File

ACCOUNTING={NO|128-255}

Set this parameter to "NO" (i.e., do not create accounting records) or to a number between 128 and 255, which specifies the SMF record type to use when writing the accounting records. In order to avoid conflicts with other applications that also produce SMF records, check with your z/OS systems programmer for an appropriate number. In addition, check with your z/OS systems programmer to ensure that the selected SMF record number is set up to be written.

Default value: NO

Retrieving Accounting Records

The standard IBM IFASMFDP utility program may be used to selectively offload Broker SMF records. Analysis and report routines - either user-written or those available from IBM or various software vendors - may subsequently be used to process the offloaded records.

```
//* Copies selected records from the "live" SMF data sets
//*
//* Replace nnn (OUTDD parameter) with a valid SMF record type
//*
//* Note: the "DISPLAY SMF" operator command will show the names of the
//* SMF data sets
//*
//IFASMFDP EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=*
//MAN1 DD DISP=SHR,DSN=SYS1.MAN1
//MAN2 DD DISP=SHR,DSN=SYS1.MAN2
//MAN3 DD DISP=SHR,DSN=SYS1.MAN3
//OUTPUT DD DISP=(MOD,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(15,15),RLSE),
// DCB=(RECFM=VBS,LRECL=32760,BLKSIZE=0),
// DSN=EXX.SMF.RECORDS
//SYSIN DD *
 DATE(2002001,2099366)
 START(0000)
 END(2359)
 INDD(MAN1,OPTIONS(DUMP))
 INDD(MAN2,OPTIONS(DUMP))
 INDD(MAN3,OPTIONS(DUMP))
 OUTDD(OUTPUT,TYPE(nnn))
//*
```

Note: The IBM publication *MVS System Management Facilities (SMF)* provides complete information on SMF.

Accounting Record Layouts

EntireX provides three mappings for its accounting records in the following members, all located in the EXX105.SRCE data set:

- EXXCACT - A C language include file that maps the accounting record;
- EXXACTR - An Assembler language MACRO that will generate a DSECT of the accounting record;
- EXXSACT - An SAS DATA step that will read in a file with the appropriate field names.

Notes

- Since there is no server for Broker Command and Information Services, no server data is generated in the SMF records for Command and Information Services conversations.
- The unit for CPUTIME is expressed in microseconds.

Example Uses of Accounting Data

Chargeback

Customers can use the EntireX accounting data to perform chargeback calculations for resource utilization in a data center. Suppose EntireX Broker is being used to dispatch messages for three business departments: Accounts Receivable, Accounts Payable, and Inventory. At the end of each month, the customer needs to determine how much of the operation and maintenance cost of EntireX Broker should be assigned to these departments. For a typical month, assume the following is true:

Department	Amount of Data	Percentage	Messages Sent	Percentage	Average Percentage
Accts Payable	50 MB	25	4000	20	22.5
Accts Receivable	40 MB	20	6000	30	25
Inventory	110 MB	55	10000	50	52.5

The use of Broker resources here is based upon both the amount of traffic sent to the Broker (bytes) as well as how often the Broker is called (messages). The average of the two percentages is used to internally bill the departments, so 52.5% of the cost of running EntireX Broker would be paid by the Inventory Department, 25% by the Accounts Receivable Department, and 22.5% by the Accounts Payable Department.

Trend Analysis

The Accounting Data can also be used for trend analysis. Suppose a customer has several point-of-sale systems in several stores throughout the United States that are tied into the corporate inventory database with EntireX. The stubs would be running at the stores, and the sales data would be transmitted to the Broker, which would hand it off to the appropriate departments in inventory. If these departments wish to ascertain when the stores are busiest, they can use the accounting data to monitor store transactions. Assume all of the stores are open every day from 9 AM to 10 PM.

Local Time	Average: Weekday Transactions per Store	Maximum Weekday Transactions in any Store	Average Weekend Transactions per Store	Maximum Weekend Transactions in any Store
9 AM	7.3	27	28.2	83
10 AM	11.2	31	29.3	102
11 AM	14.6	48	37.9	113
12 noon	56.2	106	34.8	98
1 PM	25.6	65	34.2	95
2 PM	17.2	52	38.5	102
3 PM	12.1	23	42.7	99
4 PM	18.3	34	43.2	88
5 PM	26.2	47	45.2	93
6 PM	38.2	87	40.6	105
7 PM	29.6	83	39.2	110
8 PM	18.6	78	28.6	85
9 PM	11.2	55	17.5	62

The owner of the stores can examine the data and make decisions based upon the data here. For example, on weekdays, he or she can see that there is little business until lunchtime, when the number of transactions increase. It then decreases during lunch hour; then there is another increase from 5 PM to 8 PM, after people leave work. Based on this data, the owner might investigate changing the store hours on weekdays to 10 AM to 9 PM. On the weekend the trends are different, and the store hours could be adjusted as well, although there is a more regular customer flow each hour on the weekends.

Tuning for Application Performance

Assume that a customer has two applications that perform basic request/response messaging for similar sized messages. The applications consist of many Windows PC clients and Natural RPC Servers on UNIX. An analysis of the accounting data shows the following:

Application Type	Class	Server	Service	Average Server Messages Received per Conversation	Average Client Messages Received per Conversation
Application 1:	CLASS1	SERVER1	SERVICE1	10.30	10.29
Application 2:	CLASS2	SERVER2	SERVICE2	10.30	8.98

A further analysis of the accounting data reveals that there are a lot of non-zero response codes in the records pertaining to Application 2, and that a lot of these non-zero responses indicate timeouts. With that information, the customer can address the problem by modifying the server code, or by adjusting the timeout parameters for SERVER2 so that it can have more time to get a response from the Service.

10 SSL/TLS and Certificates with EntireX

Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), are program layers for managing the security of message transmissions in a network. The idea is to contain the programming required to keep messages confidential in a program layer between an application (such as your Web browser or HTTP) and the internet's TCP/IP layers. The term sockets refers to the method of passing data back and forth between a client and a server program in a network or between program layers in the same computer. SSL and TLS use the public-and-private key encryption system from RSA, which also includes the use of a digital certificate.

This chapter describes Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Certificates within an EntireX context. The term “SSL” in this chapter refers to both SSL and TLS.

See also *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation.

Introduction

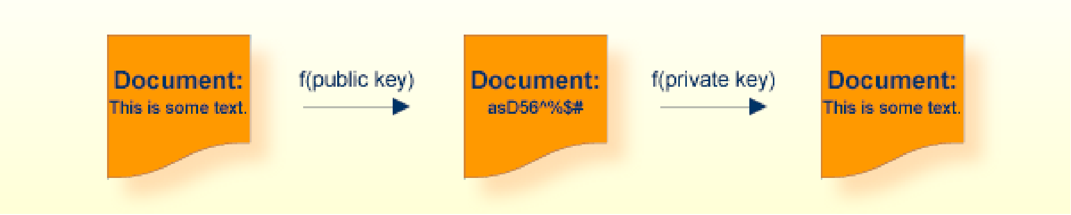
One of the major components when using SSL is the certificate. One of the tasks of certificates is to ensure that communication, which runs atop TCP/IP, adheres to an industrial-strength encryption.

Certificates can be described as electronic passports. They contain information about someone (or a machine or location), generally called the Subject. The authenticity of the subject's information is digitally signed by a trustworthy instance, called the Issuer. With certificates, this issuer is also known as a Certificate Authority (CA).

In addition to the above, a certificate also contains a random number that is called the subject's public key. Together with this public key, the subject must also be in possession of a private key. As their names suggest, the public key can be viewed by anyone, whereas the private key must be strictly secured. The public and the private keys together always form a key pair, i.e. they are always created together and complement each other.

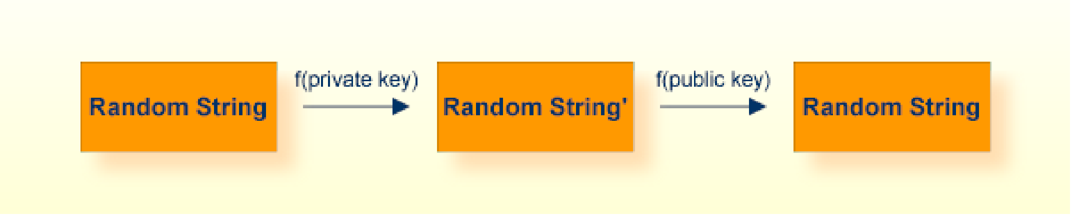
Here are some typical scenarios of their usage:

Encryption



In the image above, a public key has been used to encrypt a document. Only the owner of the private key is able to decrypt this text.

Authentication



To verify that the instance that presented a certificate is really who they claim to be (authentic), I can choose a random string, encrypt it with their public key, send it to the subject, have it decrypted with their private key and sent back. I then compare it with my original random string. Only the owner of the appropriate private key is able to perform this operation.

Random Number Generator

Another of the major components with SSL is called the Random Number Generator (RNG). To ensure genuinely random keys with each new session, SSL uses its own random number generator.

This requires a “seed”, which should be unique for each installation.

- On UNIX systems, make sure you have defined the environment variable `RANDFILE`, which refers to a file that contains at least 2048 bytes of random data. As humans are rather limited in their ability to “generate” random data, we suggest using the OpenSSL tool for this task (see *Creating Certificates with OpenSSL (z/OS, UNIX, Windows)* below).
- On Windows systems, the seed is automatically taken.

SSL/TLS Sample Certificates Delivered with EntireX

Certificates play an important role with SSL. The term “SSL” in this section refers to both SSL and TLS. In order to use SSL as the transport method for EntireX, you need to have certificates available at various locations and for various purposes. The sample certificates come as two types: a trust store (containing a public key), and a keystore (containing a private key). EntireX provides the following default certificates for preliminary test purposes:

- Default Certificates for z/OS
- Default Certificates for UNIX and Windows
- Default Certificates for Java

We strongly recommended you create your own certificates. See below for how to create your own certificates with ***OpenSSL*** and `keytool`.

Default Certificates for z/OS

After the installation process, you will find certificates in the data set `EXX105.CERT` ready to use for preliminary testing of the SSL transport:

Certificate	Description	Notes
`APPP12`	No keys can be stored directly in RACF. The pkcs12 format member `APPP12` was generated as a container for the necessary keys and the `APPCERT` member. The password to unlock this private key is `ExxAppPkcs12`.	1
`CACERT`	The CA certificate. This certificate can be used to verify the application certificate. See *Using SSL/TLS with EntireX Components*.	2

Certificate	Description	Notes
CAKEY	The private key of the CA certificate above. The password to unlock this private key is ExxCAKey. You will need this password only if you want to sign more certificates with this CA (not recommended).	
APPCERT	To be used as the SSL server certificate. If your SSL server is EntireX Broker, see SSL-specific broker attribute KEY-STORE. This certificate is signed with the private key within CAKEY.	
APPKEY	The private key of the application certificate. The password to unlock the key is ExxAppKey. If your SSL server is EntireX Broker see SSL-specific broker attributes KEY-FILE and KEY-PASSWD-ENCRYPTED.	

Notes:

1. See also the README with step-by-step description for setting up an environment that enables an SSL-secured communication with a mainframe Broker and certificates stored in RACF.
2. To allow for multiple CAs, import multiple times the various CA certificates into the keystore.

Default Certificates for UNIX and Windows

After the installation process, you will find certificates in directory *etc* ready to use for preliminary testing of the SSL transport.

Certificate	Description	Notes
ExxCACert.pem	The CA certificate. This certificate can be used to verify the application certificate. Use the SSL parameter trust_store. See *Using SSL/TLS with EntireX Components*.	1
ExxCAKey.pem	The private key of the CA certificate above. The password to unlock this private key is ExxCAKey. You will need this password only if you want to sign more certificates with this CA (not recommended).	
ExxAppCert.pem	To be used as the SSL server certificate. If your SSL server is EntireX Broker, see SSL-specific broker attribute KEY-STORE. This certificate is signed with the private key within ExxCAKey.pem	
ExxAppKey.pem	The private key of the application certificate. The password to unlock the key is ExxAppKey. If your SSL server is EntireX Broker see SSL-specific broker attributes KEY-FILE and KEY-PASSWD-ENCRYPTED.	

Notes:

1. To allow for multiple CAs, concatenate all of the CAs' .pem files into a single new .pem file.

Default Certificates for Java

After the installation process, you will find certificates in *etc* directory for preliminary testing of the SSL transport:

Certificate	Explanation	Notes
`ExxCACert.jks`	The truststore containing the default CA certificate. Use SSL parameter `trust_store`. See *Using SSL/TLS with EntireX Components*.	1
`ExxJavaAppCert.jks`	The keystore containing the application certificate. The password to unlock this container is `ExxJavaAppCert` (use SSL parameters `key_store` and `key_passwd` for Java).	

Notes:

1. To allow for multiple CAs, import multiple times the various CA certificates into the keystore.

SSL/TLS Parameters for Broker as SSL Server (One-way SSL)

The term “SSL” in this section refers to both SSL and TLS. EntireX clients and servers are always SSL clients. The SSL server can be either the EntireX Broker, EntireX Broker SSL Agent or direct RPC in webMethods Integration Server (IS inbound).

SSL usually requires a certificate on the SSL server side of a communication. In order to validate the certificate, the SSL client needs to accept the issuer of the server certificate, that is, it needs to trust the same instance that the certificate has signed. (Customs do not trust your passport - which could be forged - but instead verify its authenticity electronically!) If you are using EntireX Broker as your SSL server, use the following SSL-specific broker attributes:

Broker Attribute	Description
`KEY-STORE`	The server certificate is specified using the broker attribute `KEY-STORE`.
`KEY-FILE`	The appropriate private key is found using the broker attribute `KEY-FILE`.
`KEY-PASSWD-ENCRYPTED`	Generally, the private key is not stored in the open, it is further encrypted with a password, which - because it is often more than a single word - is sometimes also called passphrase. To use the private key properly, the application must be able to re-create the original private key. Therefore you have to provide the appropriate password with the broker attribute `KEY-PASSWD-ENCRYPTED`.

The SSL client must now present the CA (i.e. its certificate, which includes the public key), so that SSL can determine whether to accept a server certificate or not. For this purpose, specify SSL parameter `trust-store` (see below) with the EntireX client or server. Checking the SSL server certificate by an SSL client is also known as *one-way* SSL.

SSL/TLS Parameters for SSL Clients

SSL Parameter	Description
trust_store	The trust_store parameter is mandatory. It 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. This check can be disabled by specifying SSL subparameter verify_server=no.

verify_server	Possible values:
	yes Default. The common name of the server certificate (the field CN of the subject) must be equal to the Broker ID (excluding port number and transport). Example:
	`broker_id="pc001.my-company.com:1958:ssl"`
	and Broker kernel certificate (see broker attribute KEY-STORE):
	`Subject, CN=pc001.my-company.com`
	no Accept any common name (CN) in the server certificate, but still check that the certificate is signed by a trusted CA (see broker attribute TRUST-STORE).
	The default application certificate (see *SSL/TLS Sample Certificates Delivered with EntireX*) is issued to "localhost". This enables you to use a Broker ID of "localhost" together with verify_server=y.

key_store key_passwd	If the SSL server requests a client certificate (known as *two-way* SSL; verify_client=yes is defined in the configuration of the SSL server) two additional parameters have to be specified: 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.

How you provide SSL parameters depends on the EntireX component in use. See table *Using SSL/TLS with EntireX Components* below for platform and language-specific information. SSL parameters are separated by ampersand (&).

Using SSL/TLS with EntireX Components

This table provides references to available SSL documentation. Select the RPC or ACI components in use from column **SSL Client** and the communication partner such as EntireX Broker, Direct RPC, etc. from column **SSL Server**:

	SSL Client	SSL Server
	In an SSL context, SSL clients are ■ RPC clients and RPC servers ■ EntireX Adapter service and EntireX Adapter listener ■ Bridge components ■ ACI clients and ACI servers	In an SSL context, SSL servers are ■ EntireX Broker ■ EntireX Broker SSL Agent ■ *Direct RPC* in the EntireX Adapter documentation
RPC-based Components	■ For RPC clients generated by a wrapper, see *Using SSL/TLS* (C | COBOL | .NET | Java | Natural | PL/I). ■ For webMethods Integration Server, see *Support for SSL/TLS* in the EntireX Adapter documentation. ■ For RPC servers, see *Using SSL/TLS with the RPC Server* in the platform-specific administration or RPC server documentation. ■ For Bridge components, see *Using SSL/TLS* in the respective documentation section.	■ *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation ■ Broker SSL Agent in the UNIX and Windows Administration documentation ■ *Configuring Direct RPC* in the EntireX Adapter documentation
ACI-based Programming	■ For ACI clients and ACI servers, see *Using the Broker ACI with SSL/TLS* (Assembler | C | COBOL | Java | Natural | PL/I) of the programming language in use ■ For webMethods Integration Server, see *Support for SSL/TLS*	■ *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation ■ Broker SSL Agent in the UNIX and Windows Administration documentation
Administration	■ For `ETBCMD`, see *Using SSL/TLS* in section *Broker Command-line Utilities* in the respective section of the documentation ■ For `ETBINFO`, see *Using SSL/TLS* in section *Broker Command-line Utilities* in the respective section of the documentation	*Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation

SSL/TLS Certificate Creation and Handling

This section covers the following topics:

Creating Certificates with OpenSSL (z/OS, UNIX, Windows)

This section contains step-by-step instructions on how to create your own certificates. The OpenSSL tool is installed together with EntireX and can be found in directory *<install_root>/common/security/openssl/bin*.

To set up all necessary paths when working with the OpenSSL tool

- Call the installed `tlsenv` script, which is provided in the following locations:
  - Under UNIX: *<install_root>/common/security/openssl/extras/tlsenv.sh*. Source this once with the `dot` command in the POSIX shell (bash, ksh, etc.) where the OpenSSL tool will be used.
  - Under Windows: *<install_root>\common\security\openssl\extras\tlsenv.bat*. Call this once in the command line interpreter window (cmd.exe) where the OpenSSL tool will be used.

Note: Certificates adhere to a standard format and can also be created with other tools; OpenSSL is installed with EntireX and can be used as an example.

To create your own certificates

1. Create a new directory in which the new certificates will be created and where all of the other required files will be stored.
2. In your new directory create a file named *genca.cnf* with a text editor and cut and paste the contents of the file gencacnf.html (delivered with this documentation) to your new file.
3. Create an empty directory *newcerts* in your new directory.
4. Create an empty directory *certs* in your new directory.
5. Create an empty file *index.txt* in the current directory.
6. Create a file *serial* in the current directory and enter a number in column 1, line 1, for example: 1000. This serial number will be incremented for each certificate that you create.

7 Now edit the file *genca.cnf* that you cut and pasted into your new directory in step 2, above. Read the comments carefully. There are a few defaults that you will probably want to adapt to your own environment. Take care not to mix filename separators: Always use the UNIX-style forward slash "/", even on Windows.

Below is a list of the important variables that should be checked:

- Set the variable `database` to point to the *index.txt* file.
- Set the variable `serial` to point to the *serial* file.
- Set the variable `new_certs` to point to the *newcerts* directory.
- Set the variable `certs_dir` to point to the *certs* directory.
- Set the variable `certificate` to point to the CA certificate file (see *NewCACert.pem* in the example below).
- Set the variable `private_key` to point to the CA certificate's private key file (see *NewCAKey.pem* in the example below).
- Review the `req_distinguished_name` section and fill in the *_default variables, if sensible. Empty defaults will be prompted for.

8 Save the configuration file.

You can now start creating certificates.

First, you need to define a Certificate Authority (CA); create a key pair and a self-signed certificate to represent this CA.

Enter the following command in a shell and follow the instructions (be patient, loading the screen state takes several seconds)

```
openssl req -config genca.cnf -newkey rsa:4096 -x509 -keyout <NewCAKey.pem> -out ↩
<NewCACert.pem> -days 365
```

Do not forget the passphrase for the key file! You will need it whenever a new certificate is generated.

Now you have a CA certificate and a CA key file.

Next, create a certificate that can be used by various products (for example the Broker kernel) to start an SSL server session.

With the CA cert and key files described above you can create any number of certificates. We will sign all of them with the same CA (used from the *genca.cnf* file).

Create a certificate request:

```
openssl req -config genca.cnf -newkey rsa:2048 -out <ExxAppCertReq.pem> -keyout ↩
<ExxAppKey.pem> -days 365
```

You will be prompted for a new passphrase. Again, this will be the passphrase to lock the *MyAppKey.pem* file. Remember it well.

You must then sign this certificate request with your CA to create a proper certificate:

```
openssl ca -config genca.cnf -policy policy_anything -out <ExxAppCert.pem> -infiles ↩
<ExxAppCertReq.pem>
```

Note: The passphrase you are prompted with is the one used to unlock the CA key.

Creating Certificates with keytool (Java)

A certificate management tool is also supplied with the standard JDK kit, i.e. it is part of J2SE kit, not the JSSE kit. Certificate requests can be generated and keystores and truststores can be built with this tool. The steps for building keystores and truststores are outlined below.

To create a keystore

1 Create a keystore containing a self-signed certificate and key (example yourkeystore).

The following command will prompt you for identification information.

```
 keytool -genkey -v -alias yourJavaApp -keyalg RSA -validity 900 -keypass ↩
yourkeypsw -keystore yourkeystore -storepass  yourkeypsw
```

2 Import any CA certificates of CAs which will sign the certificate generated above.

```
keytool -import -v -alias yourcacert -file yourcacert.pem -keystore yourkeystore ↩
-storepass yourkeypsw
```

3 (Optional) List certificate chain present in keystore.

```
keytool -list -v -keystore yourkeystore -storepass yourkeypsw
```

4 Extract certificate for signing by a CA.

```
keytool -certreq -v -alias yourJavaApp -file yourJavaAppreq -keypass yourkeypsw ↩
-keystore yourkeystore -storepass yourkeypsw
```

5 Sign Java certificate request with OpenSSL tool.

```
openssl ca -config yourca.cnf -policy policy_anything -out yourjavaapp.pem ↩
-notext -days 365 -infiles yourJavaAppreq
```

Note: The `-notext` parameter is required. Without it, the import of a signed certificate to keystore will fail. The error will be either a `Not an X.509 certificate` or a `Tag sequence` error. The reason for the error is that the OpenSSL signing tool will write both a text version and an encoded version of the signed certificate to the output file if the `-notext` parameter is not specified.

6 Import signed certificate.

```
keytool -import -v -alias yourJavaApp -file yourjavaapp.pem -keypass yourkeypsw ↩
-keystore yourkeystore -storepass yourkeypsw
```

Notes:

1. *yourjavaapp.pem* is the signed certificate returned by the CA.
2. Import will only work if a signed CA certificate is already present in the keystore.

To create a truststore

- Import the CA certificates that were used to sign the client and server certificates.
  - Import signed CA certificates.

    ```
    keytool -import -v -alias yourcacert -file yourcacert.pem -keystore ↩
    yourtruststore -storepass yourstorepsw
    ```

  - (Optional) List truststore.

    ```
    keytool -list -v -keystore yourtruststore -storepass yourstorepsw
    ```

Importing Certificates into RACF (z/OS)

This section applies to operating system z/OS only.

To import certificates into RACF

1. Create a certificate with OpenSSL. See *Creating Certificates with OpenSSL (z/OS, UNIX, Windows)*.
2. Create the PKCS#12 import format for RACF. Enter the following command to create a file containing the application certificate and application key files for import into RACF:

   ```
   openssl pkcs12 -export -inkey <EXXAppKey.pem> -in <EXXAppCert.pem> -certfile ↩
   <EXXCACert.pem> -out <EXXPkcs12.p12>
   ```

   You will be prompted for the passphrase of the private key and for an export password. The output file is created in PKCS#12 format. You can use FTP to transfer the output file in binary mode to the IBM host.
3. Import certificates and private keys with `RACDCERT` into RACF. See readme file EXX105.CERT(README) in the product distribution for detailed instructions.

Additional Considerations for PKI (Public Key Infrastructure)

When using a PKI, there are usually more than two certificates involved. Typically, there is one (self-signed) root certificate, one or more CA certificates, and several application certificates, usually one for every server.

For the SSL server side (Broker) you need a suitable application certificate.

To check the certificate

- Execute the command:

  ```
  openssl x509 -in <YourSSLCert.pem> -text
  ```

  This will display relevant information about the certificate such as key extensions with key usage and basic constraints. (For example, the Basic Contraint CA should be "FALSE".)

Given a specific server certificate, it is also possible to verify the certificate chain.

To verify the certificate chain

- Execute the command:

```
openssl verify -CAfile <YourCaCert.pem> -purpose sslserver <YourSSLCert.pem>
```

If you receive an OK, then `<YourSSLCert.pem>` should work on the SSL server side together with the `<YourCaCert.pem>` on the SSL client side.

Note: If there is a chain of CA certificates defined, copy the contents of the appropriate CA*xxx*.pem files into one new file and use this as the `<YourCaCert.pem>` on the client side to verify the SSL server certificate against.

Support of Self-signed Certificates

To support self-signed certificates it is probably necessary to modify the LDAP settings. For example, to allow use of a self-signed certificate in OpenLDAP, the client needs access to the CA's certificate. Add the following line to file */etc/openldap/ldap.conf*:

```
TLS_CACERT <YourCaCert.pem>
```

11 Authorization Rules

An authorization rule is used to perform access checks for authenticated user IDs against lists of services defined within the rule. This feature is available on UNIX and Windows using EntireX Security on these platforms. Authorization rules can be stored in the Broker attribute file or in an LDAP repository.

Introduction

The value of `SECURITY-SYSTEM` in the `DEFAULTS=SECURITY` section of the Broker attribute file determines the location of the authorization rules:

- **Broker Attribute File**
  Set `SECURITY-SYSTEM=OS`.
  Rules are defined under `DEFAULTS=AUTHORIZATION-RULES` of the broker attribute file.
- **LDAP Repository**
  Set `SECURITY-SYSTEM=LDAP`.
  Rules are stored in an LDAP repository. Security-specific attributes `LDAP-AUTHENTICATION-URL` and `LDAP-AUTHORIZATION-URL` define the parameters for the access of the LDAP client side, and `LDAP-AUTHORIZATION-RULE` defines applicable rule names.

Whenever an authorization call occurs, the Broker security exit performs checks based on the value of the security-specific attribute `AUTHORIZATION-DEFAULT`. Examples of these two approaches are provided below.

Rules Stored in Broker Attribute File

Set `SECURITY-SYSTEM=OS` in the `SECURITY-SYSTEM` section of the broker attribute file and define the individual rules under `DEFAULTS=AUTHORIZATION-RULES`. A rule is a container for a list of services and a list of client and server user IDs. All users defined in a rule are authorized to use all services defined in this rule.

Sample Attribute File Settings

```
DEFAULTS=SECURITY
   SECURITY-SYSTEM = OS
   SECURITY-LEVEL = AUTHORIZATION
   AUTHORIZATION-DEFAULT = NO

DEFAULTS = AUTHORIZATION-RULES
   RULE-NAME = rule1
      CLASS = class1, SERVER = server1, SERVICE = service1
      CLIENT-USER-ID = user1
      CLIENT-USER-ID = user2
      SERVER-USER-ID = user3
      SERVER-USER-ID = user4
   RULE-NAME = rule2
      CLASS = class2, SERVER = server2, SERVICE = service2
      CLASS = class3, SERVER = server3, SERVICE = service3
      CLIENT-USER-ID = user1
      CLIENT-USER-ID = user5
      CLIENT-USER-ID = user6
      SERVER-USER-ID = user7
```

This example results in the following permissions:

- `user1` may send requests to all three services.
- `user2` may send requests to `service1` only.
- `user5` and `user6` may send requests to `service2` and `service3`, but not `service1`.
- `user3` and `user4` may run as servers of `service1`.
- `user7` may run as server of `service2` and `service3`.

Attributes are described in more detail under *Security-specific Attributes* and *Authorization Rule-specific Attributes*.

Rules Stored in LDAP Repository

This section covers the following topics:

- Sample Attribute File Settings
- Configuring your LDAP Repository
- Authorization Rule Data
- Hints for Microsoft Active Directory

Sample Attribute File Settings

Specify the URL of your LDAP server under `LDAP-AUTHENTICATION-URL` and `LDAP-AUTHORIZATION-URL` in the `DEFAULTS=SECURITY` section of the broker attribute file, and specify up to 16 rules with `LDAP-AUTHORIZATION-RULE` as shown in the example below:

```
DEFAULTS=SECURITY
   SECURITY-SYSTEM = LDAP
   SECURITY-LEVEL = AUTHORIZATION
   LDAP-AUTHENTICATION-URL = "ldap://myhost.mydomain.com"
   LDAP-AUTHORIZATION-URL  = "ldap://myhost.mydomain.com"
   LDAP-AUTHORIZATION-RULE = rule1
   LDAP-AUTHORIZATION-RULE = rule2
   ...
   LDAP-AUTHORIZATION-RULE = rule16
   LDAP-PERSON-BASE-BINDDN   = "cn=users,dc=software-ag,dc=de"
   LDAP-SASL-AUTHENTICATION  = YES
```

Note: We assume you can change authorization rules (add/modify/delete) in LDAP directly. Add/delete authorization rule names in Broker attribute file accordingly.

Attributes are described in more detail under *Security-specific Attributes*.

Configuring your LDAP Repository

An LDAP server is a prerequisite (based on LDAPv3); it is not installed with EntireX.

For the installation of the LDAP server, see the respective product documentation. All servers have to support the attribute types `sag-key`, `sag-value` and the objectClass `sag-xds`. They are defined in the following schema.

```
attributetypes:
      ( 1.2.276.0.12.2.1.1
      NAME 'sag-key'
      DESC 'User Defined Attribute'
      SYNTAX '1.3.6.1.4.1.1466.115.121.1.26')
attributetypes:
      ( 1.2.276.0.12.2.1.2
      NAME 'sag-value'
      DESC 'User Defined Attribute'
      SYNTAX '1.3.6.1.4.1.1466.115.121.1.5')
objectclasses:
      ( 1.2.276.0.12.2.3.1
      NAME 'sag-xds'
      DESC 'User Defined ObjectClass'
      SUP 'top'
      MUST ( objectclass $ sag-key )
      MAY ( aci $ sag-value ) )
```

We recommend setting up a separate branch in the directory for authorization rules. The distinguished name of this branch is the value of the configuration setting specified with attribute `LDAP-BASE-DN` in section *Security-specific Attributes* in the platform-independent administration documentation.

Authorization Rule Data

The following example describes the required data in LDAP to define the authorization rule RULE1 restricting service SC1:SN1:SV1 (CLASS=SC1, SERVER=SN1,SERVICE=SV1) to authorized client CLIENT1 and authorized server SERVER1. It assumes attribute LDAP-BASE-DN was set to "dc=software-ag,dc=de".

Define the authorization rule:

```
sag-key=RULE1,sag-key=100,sag-key=AuthRules,sag-key=EntireX,sag-key=Software ↩
AG,dc=software-ag,dc=de
```

Define the service for the authorization rule:

```
sag-key=SC1:SN1:SV1,sag-key=RULE1,sag-key=100,sag-key=AuthRules,sag-key=EntireX,sag-key=Software ↩
AG,dc=software-ag,dc=de
```

Define a client user ID for the service:

```
sag-key=CLIENT1 ↩
[C,sag-key=SC1:SN1:SV1,sag-key=RULE1,sag-key=100,sag-key=AuthRules,sag-key=EntireX,sag-key=Software ↩
AG,dc=software-ag,dc=de
```

Define a server user ID for the service:

```
sag-key=SERVER1 ↩
[S,sag-key=SC1:SN1:SV1,sag-key=RULE1,sag-key=100,sag-key=AuthRules,sag-key=EntireX,sag-key=Software ↩
AG,dc=software-ag,dc=de
```

The part "sag-key=100,sag-key=AuthRules,sag-key=EntireX,sag-key=Software AG" identifies authorization rules in general. All values are fixed and must not be changed. Preceeding "sag-key=RULE1" defines the name of an authorization rule. This rule name must have been defined with attribute LDAP-AUTHORIZATION-RULE in the Broker attribute file.

The definition of services requires "sag-key=SC1:SN1:SV1" in front of the complete rule data.

User ID values contain the user ID plus blank, open square bracket and uppercase C for clients or S for servers.

Following table lists attribute type and value. All entries belong to objectClass sag-xds.

Attribute Type	Value
sag-key	Software AG
sag-key	EntireX
sag-key	AuthRules
sag-key	100
sag-key	RULE1
sag-key	SC1:SN1:SV1
sag-key	CLIENT [C
sag-key	SERVER [S

Hints for Microsoft Active Directory

Note: To deploy the `sagxds` schema on Microsoft Active Directory, do not use the Microsoft Active Directory tools for editing the schema. Use the following step-by-step instructions:

To deploy the `sagxds` schema

1 Make a backup of the system state. Changes to the schema of Microsoft Active Directory are irreversible without a backup of the system state.

2 You must enable UPDATE schema.

1. To make the Schema Master available, enter the following at a command prompt:

```
regsvr32.exe schmmgmt.dll
```

2. Enter MMC.
3. From Console menu item, select: **Add/remove snap-in**.
4. Choose **Add**.
5. Choose **Active Directory Schema** from **Action** menu item of Active Directory Schema, select **Operations Master**.
6. Choose “The schema may be modified on this domain controller”.

3 Copy the following text to the file *sagxds.ldif*

```
# Add sag-value attribute

dn: CN=sag-value,CN=Schema,CN=Configuration,DC=<your domains name>
changetype: add
adminDisplayName: sag-value
attributeID: 1.2.276.0.12.2.1.2
attributeSyntax: 2.5.5.10
cn: sag-value
isSingleValued: FALSE
lDAPDisplayName: sag-value
distinguishedName: CN=sag-value,CN=Schema,CN=Configuration,DC=<your domains name>
objectCategory:
 CN=Attribute-Schema,CN=Schema,CN=Configuration,DC=<your domains name>
objectClass: attributeSchema
oMSyntax: 4
name: sag-value

# Add sag-key attribute
# Active Directory requires the naming attribute(RDN) to be a syntax of ↩
DirectoryString

dn: CN=sag-key,CN=Schema,CN=Configuration,DC=<your domains name>
changetype: add
adminDisplayName: sag-key
attributeID: 1.2.276.0.12.2.1.1
attributeSyntax: 2.5.5.12
cn: sag-key
isMemberOfPartialAttributeSet: TRUE
isSingleValued: TRUE
lDAPDisplayName: sag-key
distinguishedName: CN=sag-key,CN=Schema,CN=Configuration,DC=<your domains name>
objectCategory:
 CN=Attribute-Schema,CN=Schema,CN=Configuration,DC=<your domains name>
objectClass: attributeSchema
oMSyntax: 64
name: sag-key
searchFlags: 1

# Update the schema

DN:
changetype: modify
add: schemaUpdateNow
schemaUpdateNow: 1
-

# Add sag-xds class

dn: CN=sag-xds,CN=Schema,CN=Configuration,DC=<your domains name>
changetype: add
adminDescription: sag-xds
adminDisplayName: sag-xds
```

```
cn: sag-xds
defaultObjectCategory:
 CN=sag-xds,CN=Schema,CN=Configuration,DC=<your domains name>
governsID: 1.2.276.0.12.2.3.1
lDAPDisplayName: sag-xds
mayContain: sag-value
mustContain: sag-key
distinguishedName: CN=sag-xds,CN=Schema,CN=Configuration,DC=<your domains name>
objectCategory: CN=Class-Schema,CN=Schema,CN=Configuration,DC=<your domains name>
objectClass: classSchema
objectClassCategory: 1
possSuperiors: container
name: sag-xds
rDNAttID: sag-key
subClassOf: top

# Update the schema

DN:
changetype: modify
add: schemaUpdateNow
schemaUpdateNow: 1
-

# Modify sag-xds class
# make sag-xds a possSuperior. This means a sag-xds class can contain other ↩
sag-xds classes.

dn: CN=sag-xds,CN=Schema,CN=Configuration,DC=<your domains name>
changetype: modify
add: possSuperiors
possSuperiors: sag-xds
-

# Update the schema

DN:
changetype: modify
add: schemaUpdateNow
schemaUpdateNow: 1
-
```

4 Replace all instances of `dc= <your domain name>` with your domain name, for example `dc=myunit,dc=mycompany,dc=com`.

5 Run it with the command:

```
ldifde -s <your server> -b <account> <domain> <password> -i -f sagxds.ldif
```

6 Add containers that represent the base DN of the authorization rules. These containers determine the value of attribute `LDAP-BASE-DN` under *Broker Attributes*. Example (for two containers):

```
dn: CN=<your container 1>,DC=<your domain name>
changetype: add
cn: <your container 1>
objectclass: container

dn: CN=<your container2>,<your container 1>,DC=  <your domain name>
changetype: add
cn: <your container 2>
objectclass: container
```

7 With the utilities for Microsoft Active Directory, set the permissions to read and to modify the containers.

12 Data Compression in EntireX Broker

Data compression within EntireX Broker allows you to exchange smaller packet sizes between clients and servers. This helps to reduce response time during transmissions as well as improve the overall network throughput, especially with low-bandwidth connections.

This chapter gives an overview of data compression in EntireX Broker.

See also: COMPRESSLEVEL under *Broker ACI Fields* | *Data Compression* under *Writing Client and Server Applications* in the ACI Programming documentation.

Introduction

Compression is performed only on the SEND and RECEIVE buffers. The client or server application has the option of setting the level of compression/decompression for data transmission. The compression level can be set to achieve either no compression or a range of compression/decompression. If during a data transmission the data buffer does not compress, a logged warning message 00200450 indicates that the data has not been compressed during transmission.

Note: The compression level is used to control compression only between the application and the Broker kernel.

zlib

zlib is a general-purpose software implementing data compression across a variety of platforms. Version 1.1.4 of zlib is implemented starting with EntireX Broker version 7. The functions used within EntireX Broker represent a subset of those available within the zlib software.

The compression algorithms are implemented through the open source software **zlib**.

Implementation

Compression of the data is implemented by the following components of EntireX:

<table>
<tr><th>Components</th><th>Description</th></tr>
<tr><td>Broker control block</td><td>The Broker control block (ETBCB) contains a field that is used to set the compression level. This field determines for any <code>SEND</code>/<code>RECEIVE</code> transmission whether the data buffer will be compressed/decompressed. Possible values:
<table>
<tr><th>0 - 9</th><td>0 = no compression, 9 = maximum compression/decompression</td></tr>
<tr><th>N</th><td>Default. No compression.</td></tr>
<tr><th>Y</th><td>Compression level 6</td></tr>
</table>
If the data buffer does not compress, the kernel or stub generates a logged warning message 00200450 indicating that the transmitted data is not compressed.

<b>Note:</b> See also ACI control block field <code>COMPRESSLEVEL</code>.</td></tr>
<tr><td>Stubs: Broker stub and Java stub</td><td>The behavior of the Broker stub and Java stub is identical with respect to compression.

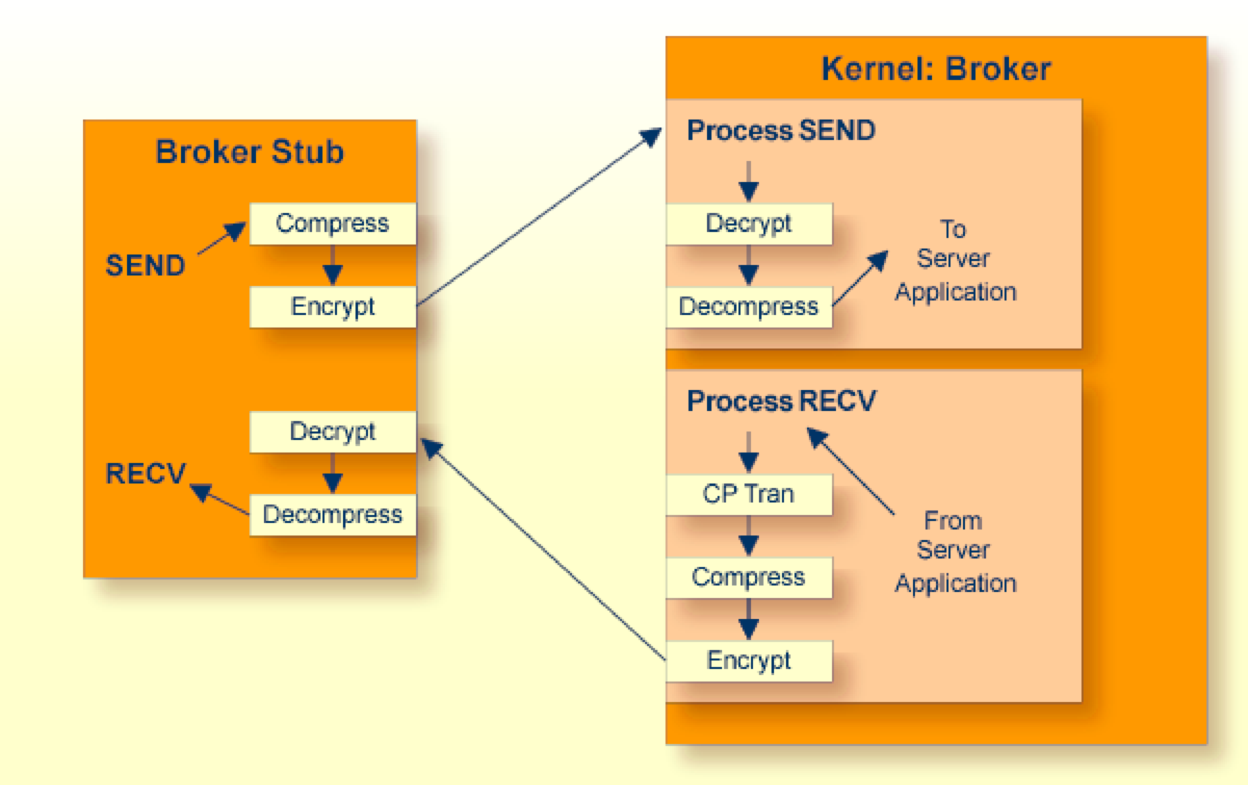
The logic of a client or server application sets the compress level of the Broker control block when it issues the <code>SEND</code> or <code>RECEIVE</code> command. If the application issues a <code>SEND</code>, the stub compresses the data buffer before transmission of the data. If the application issues a <code>RECEIVE</code>, the stub decompresses the data buffer after reception of the data.

<b>Note:</b> The compression level is used to control compression only between the application and the Broker kernel.</td></tr>
<tr><td>Broker kernel</td><td>When a client or server application <code>SEND</code>s the data to the Broker kernel, the application specifies the level at which the kernel is to decompress the data.

When the client or server application issues the <code>RECEIVE</code> command, the Broker kernel compresses the data before returning it to the application. The application specifies the level at which the kernel is to compress the data.</td></tr>
</table>

Sequencing Summary

The following graphic shows the sequencing of data compression within EntireX Broker:



Sample Programs

Using the `-rn` option will cause compression to be used at level *n*.

- `bcoc` can be instructed to use compression/decompression by specifying, for example:

  ```
  bcoc -r2
  ```

  This will cause a compression/decompression level of 2 to be used on all transmissions between the client and the broker.

- `bcos` can be instructed to use compression/decompression by specifying, for example:

```
bcos -r4
```

This will cause a compression/decompression level of 4 to be used on all transmissions between the server and the broker.

To test how well various types of data will compress, you can use the option `-g`*`filename`*. You can use, for example, the following syntax to specify that input is to be from a pre-existing file, using the following arguments:

-
  ```
  bcoc -r2 -gmyfile1.txt
  ```

  This will read in *myfile1.txt* and send it to a registered server. If `bcos` is the server, `bcos` will reverse the data sequence and return the data.

-
  ```
  bcos -r4 -gmyfile2.txt
  ```

  This will write in *myfile2.txt* the data sent from the client.

13 Timeout Considerations for EntireX Broker

This chapter describes the timeout settings for EntireX Broker.

Timeout Units

The timeout duration can be specified in seconds (S), minutes (M) or hours (H), for example 100M. If no unit is specified, the default is seconds.

Timeout Settings

Timeout Setting	Description
Client Non-activity Timeout	Any broker stub application that issues a `LOGON` but does not issue a `REGISTER` is a client. During logon, broker allocates resources to each client and keeps them available to the client until the client application issues a `LOGOFF`. A client is considered inactive when it is not issuing a broker request. A typical example of a broker request by a client is the `SEND` function. The `CLIENT-NONACT` value defines the maximum period of time a client can remain inactive. See `CLIENT-NONACT` under *Broker Attributes*. If the client continues to be inactive beyond this period of time, Broker releases all the resources allocated to this client. This time is a global attribute, applicable to all clients of the Broker.
Server Non-activity Timeout	Any broker stub application that issues a `LOGON` and also issues a `REGISTER` is a server. During logon and registration, broker allocates resources to each server, and keeps them available to the server until the server issues a `DEREGISTER` and `LOGOFF`. A server is considered inactive when it is not issuing a broker request. A typical example of a Broker request by a server is the `RECEIVE` function. The `SERVER-NONACT` value defines the maximum period of time a server can remain inactive. See `SERVER-NONACT` under *Broker Attributes*. If the server continues to be inactive beyond this period of time, Broker releases all the resources allocated to this server. This time is a per-service attribute, and can vary from one service definition to another. All servers, registered to the same service, inherit the same `SERVER-NONACT` time. If a server registers to more than one service, the highest `SERVER-NONACT` value is taken as the non-activity time period.
Conversation Non-activity Timeout	A conversation begins when a client successfully sends a message addressed to a server. The Broker allocates a unique conversation, even before the server receives this message. Broker also allocates resources to manage each conversation. A conversation remains active as long as messages are being exchanged with this conversation ID. The conversation remains inactive as long as neither a client nor a server makes a Broker request, referencing this conversation ID. The resources allocated to a conversation are freed when either a client or a server issues `EOC`.

Timeout Setting	Description
	The CONV-NONACT value defines the maximum period of time a conversation can remain inactive. If the conversation continues to be inactive beyond this period of time, Broker releases all the resources allocated to this conversation.
UOW Lifetime (UWTIME)	Each UOW has a lifetime value associated with it. This is the time that a UOW is allowed to exist without being completed. A UOW is completed when it is successfully ■ either cancelled or backed out by its sender ■ or cancelled or committed by its receiver. If a UOW is in ACCEPTED status when this lifetime expires, the UOW is placed into a timeout status. Lifetime timeouts will not occur when the UOW is in either RECEIVED or DELIVERED status. See CONV-NONACT description in ***Relationship between Timeout Values***.
Transport Timeouts	If Entire Net-Work is used to transmit a Broker request, the setting of the Entire Net-Work NODE statement parameter REPLYTIM may influence the behavior of the application (see your Entire Net-Work documentation for details). All non-activity timeouts in the Broker configuration should be considered when determining the maximum time. This maximum time should be less than the value defined for REPLYTIM in the Entire Net-Work configuration.

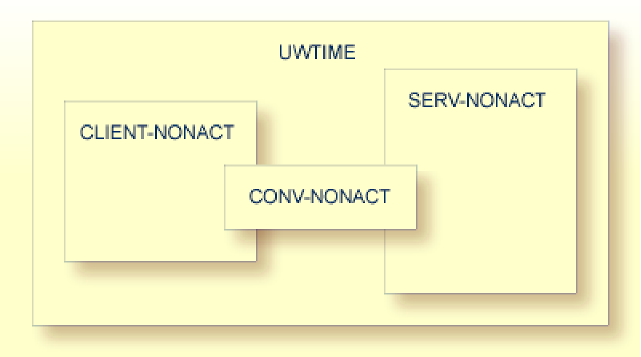
Relationship between Timeout Values

The interdependency between different timeouts is described as follows:

- UOW Messages

- Non-UOW Messages

UOW Messages



- A server or a client engaged in a conversation will not be timed out until the UOW that they are handling times out. `CLIENT-NONACT` (or `SERV-NONACT`) has no effect if it is shorter than `UWTIME`.
- A conversation may time out earlier than either the client or the server. When an existing conversation times out, the participating server and client can start a new conversation. We recommend you set the `CONV-NONACT` shorter than `CLIENT-NONACT` (or `SERV-NONACT`).
- If either the client or server times out before the conversation does, the conversation does not continue, that is, it reaches end of conversation (EOC). Nevertheless, the surviving participant (client or server) can continue and receive any unread messages.
- When a conversation times out, Broker checks for the status of all UOWs in this conversation. Any UOW with status `RECEIVED` or `DELIVERED` is backed out and enters into `ACCEPTED` status. "Accepted" means that the UOW can be received by anyone (with `CONV-ID=NEW`), and that the conversation has lost the link to the consumer of the UOW.

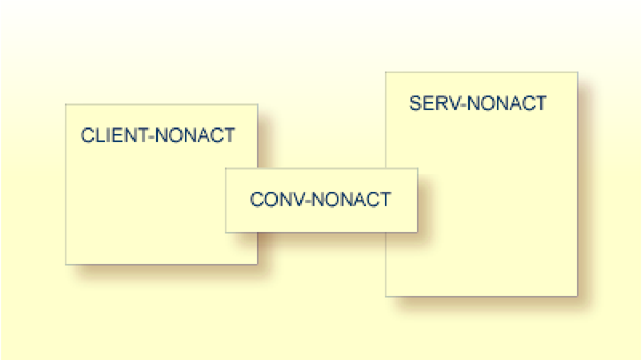
  Note: The link to the consumer is lost only for the first UOW in a conversation when the status changes to `ACCEPTED`; with subsequent UOWs, the link is not lost.

- A common relationship between these three timeout values is as follows, although this may not be the optimum combination in all situations:

  `UWTIME` > `SERV-NONACT` > `CLIENT-NONACT` > `CONV-NONACT`

  In common situations, this combination will achieve optimal resource consumption without recourse to repeatedly restarting applications.

Non-UOW Messages



Timeout behavior remains the same as in UOW messages, except that UWTIME (UOW lifetime attribute) is not applicable here. The optimal hierarchy between the three timeout values is shown below:

SERV-NONACT > CLIENT-NONACT > CONV-NONACT

Timeout-related Error Messages

When any client or server or conversation times out, the Broker does not immediately notify the application. The application receives notification when it makes its next Broker request. The following are the error messages commonly associated with the respective timeouts. The errors listed below can occur in the case of blocked and non-blocked ACI calls. A blocked call is one in which the ACI field WAIT is set to either "YES" or a non-zero numeric value.

See message 00740074.

- CLIENT-NONACT
- SERV-NONACT
- CONV-NONACT
- Special Case for UOW Messages

CLIENT-NONACT

In the following errors, it is assumed that client only has timed out, while the server and conversation are active.

Error Number	Error Text	Explanation
00020002	User does not exist	When the timed out client tries to make a Broker request.
00030012	EOC due to LOGOFF of partner	The surviving partner (server) receives this error when attempting to receive on a conversation which is closed because the client has timed out. If there are any unread messages, the server successfully receives them.

SERV-NONACT

In the following errors, it is assumed that only the server has timed out, while the client and conversation are active.

Error Number	Error Text	Explanation
00020002	User does not exist	When the timed out client tries to make a Broker request.
00030067	Partner timeout occurred	The surviving partner (client) receives this error when attempting to send on a conversation which is closed because the server timed out.

CONV-NONACT

It is assumed that server and client are active.

Error Number	Error Text	Explanation
00030003	No matching conversation found	When either a server or a client attempts a new Broker request affecting this timed out conversation.
00030073	Conversation timeout occurred	When both client and server are already engaged in a conversation, and the conversation time out without the partner issuing any Broker request.

Special Case for UOW Messages

UOWs involved in a conversation, and which are in `DELIVERED` state, revert to `ACCEPTED` state when the conversation times out. UOWs in `ACCEPTED` state are no longer bound to a server nor to an existing conversation. Therefore, UOW in `ACCEPTED` state is part of a new conversation that is available to any server.

14 EXXMSG - Command-line Tool for Displaying Error Messages

EXXMSG is a command-line tool that displays the text of an EntireX error message for a supplied error number. It is available on all platforms.

Running the EXXMSG Command-line Utility

Under z/OS, command-line utility EXXMSG is located in library EXB105.LOAD. Under UNIX and Windows, the utility is located in the EntireX *bin* directory.

Command-line Parameters

The only command-line parameter is any 8-digit error code.

Sample Command

```
exxmsg 02150148
```

Sample Output

```
Software AG webMethods EntireX 9.0.0 (473) Linux 3.1.10-1.16-desktop
(c) Copyright 1997 - 2012 Software AG. All rights reserved.

02150148    EntireX Broker not active : (or Transport-Specific Error Text)
Explanation The requested Broker specified in BROKER-ID is not reachable.
Action      Check the BROKER-ID. If it is correct, check if ETB_TRANSPORT
            environment variable is defined and if defined, it should point to
            the desired transport method. If problem persists, contact your
            network administrator.
```

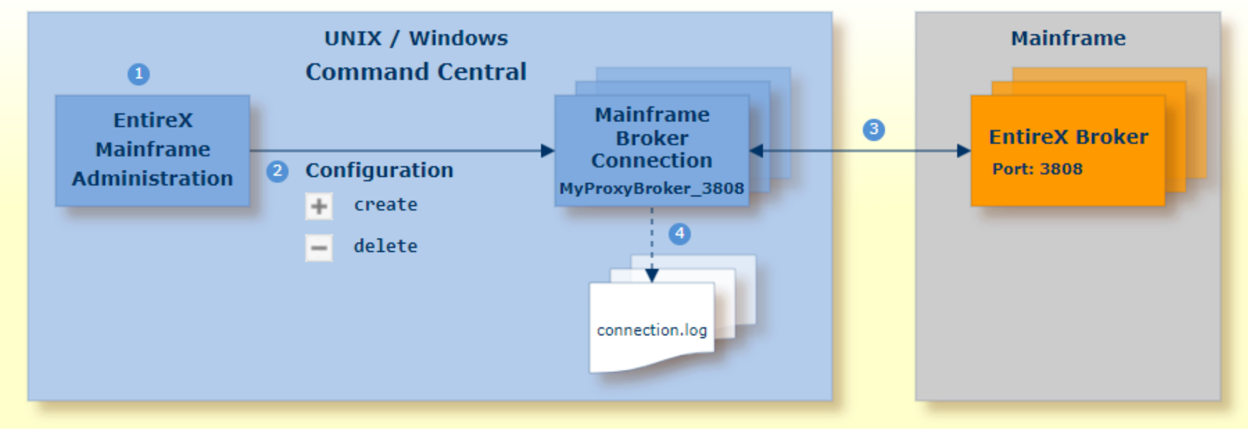
15 Introduction to Monitoring EntireX Mainframe Broker using Command Central

See also *Monitoring EntireX Mainframe Broker* using the Command Central GUI | Command Line.

Note: Command Central functionality that is not EntireX-specific is described in the separate Command Central documentation or the online help provided with Command Central. On Empower, the documentation is provided under webMethods > EntireX > EntireX 10.5 > Additional Documentation.

Scope

This section applies to Broker instances running on mainframe platforms z/OS, BS2000 and z/VSE. For EntireX Brokers running on UNIX and Windows platforms, see *Introduction to Administering EntireX Broker with Command Central (UNIX and Windows)*.



The EntireX Mainframe Administration instance is automatically provided in Command Central. For more information see the separate Command Central documentation or the online help provided with Command Central.

From the **Configuration** tab (GUI) or with the command-line interface you can create or delete Mainframe Broker Connections to a broker running in a mainframe environment.

Use Mainframe Broker Connections to perform the following operations on EntireX Broker:

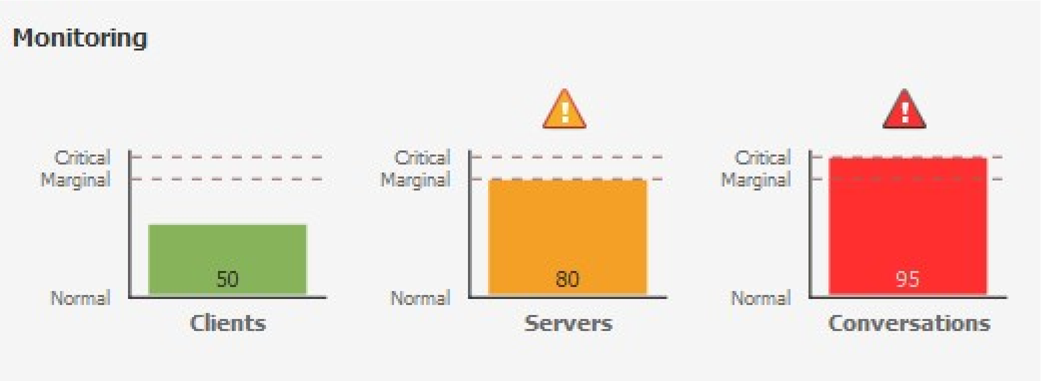
- view the EntireX Brokers running in each environment of your IT landscape
- monitor runtime status, KPIs (key performance indicators), and alerts of EntireX Broker instances
- display services
- display server instances of a service

When you create a mainframe connection, this is logged to file *connection.log* of the mainframe connection instance.

Note: Do not confuse this logfile with the broker log on the mainframe (for example `DD:SYSOUT` under z/OS).

Monitoring EntireX Broker KPIs

The visual key performance indicators (KPIs) and alerts enable you to monitor webMethods EntireX Broker's health. The following KPIs help you administer, troubleshoot, and resolve performance issues in EntireX Broker:



KPI	Description
Clients	Number of active clients.
Servers	Number of active servers.
Conversations	Number of active conversations.

Supported Configuration Types

Command Central supports the following configuration instance:

Instance	Type	Use to...
`EXX-MONITORING-KPIS`	`EXX-MONITORING-KPIS`	Show and edit the Monitoring KPI settings, like the marginal and critical bounds, etc.

16 Monitoring EntireX Mainframe Broker using the Command Central GUI

See also *Monitoring EntireX Mainframe Broker* and *Monitoring EntireX Mainframe Broker using the Command Central Command Line*.

Logging in to Command Central

To log in to Command Central

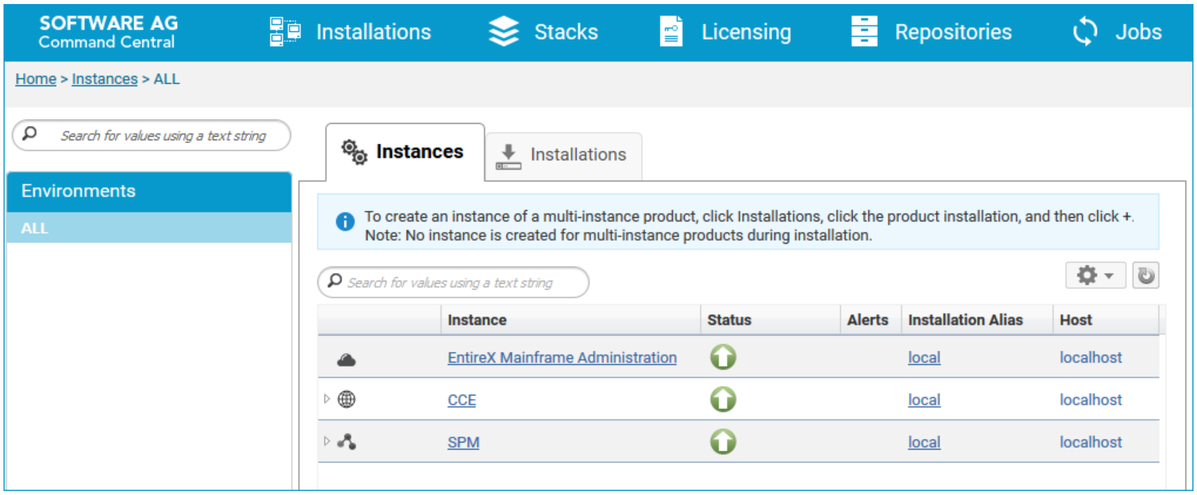
1 Open an Internet browser and specify the URL of the Command Central Server: “http://<`Command_Central_host`>:<`Command_Central_port`>”.

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.

2 In the **Login** page, provide your user credentials and click **Log In**.

This takes you to the page **Home > Instances**:



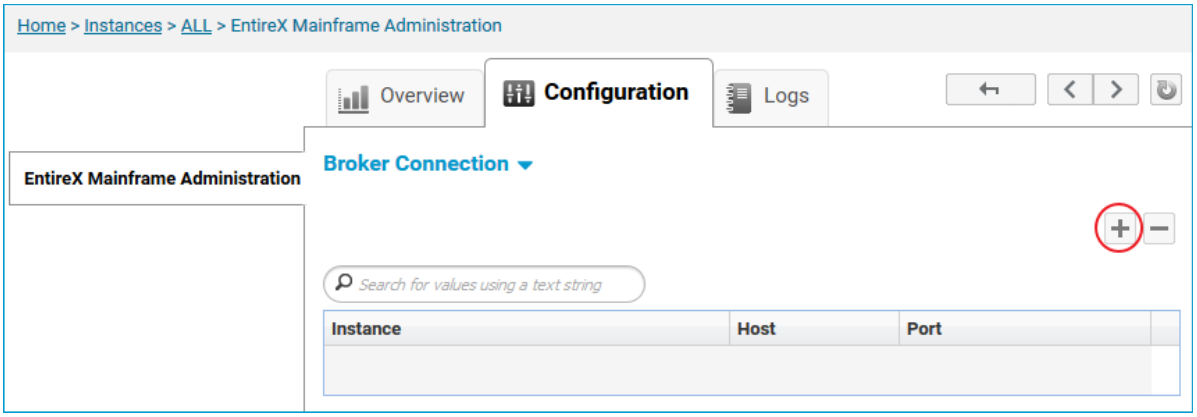
To create an instance of a multi-instance product, click Installations, click the product installation, and then click +.
Note: No instance is created for multi-instance products during installation.

	Instance	Status	Alerts	Installation Alias	Host
	EntireX Mainframe Administration	⬆		local	localhost
	CCE	⬆		local	localhost
	SPM	⬆		local	localhost

Creating an EntireX Mainframe Broker Connection

To create an EntireX Mainframe Broker connection

1 Navigate to **Home > Instances > ALL > EntireX Mainframe Administration** and click the **Configuration** tab.

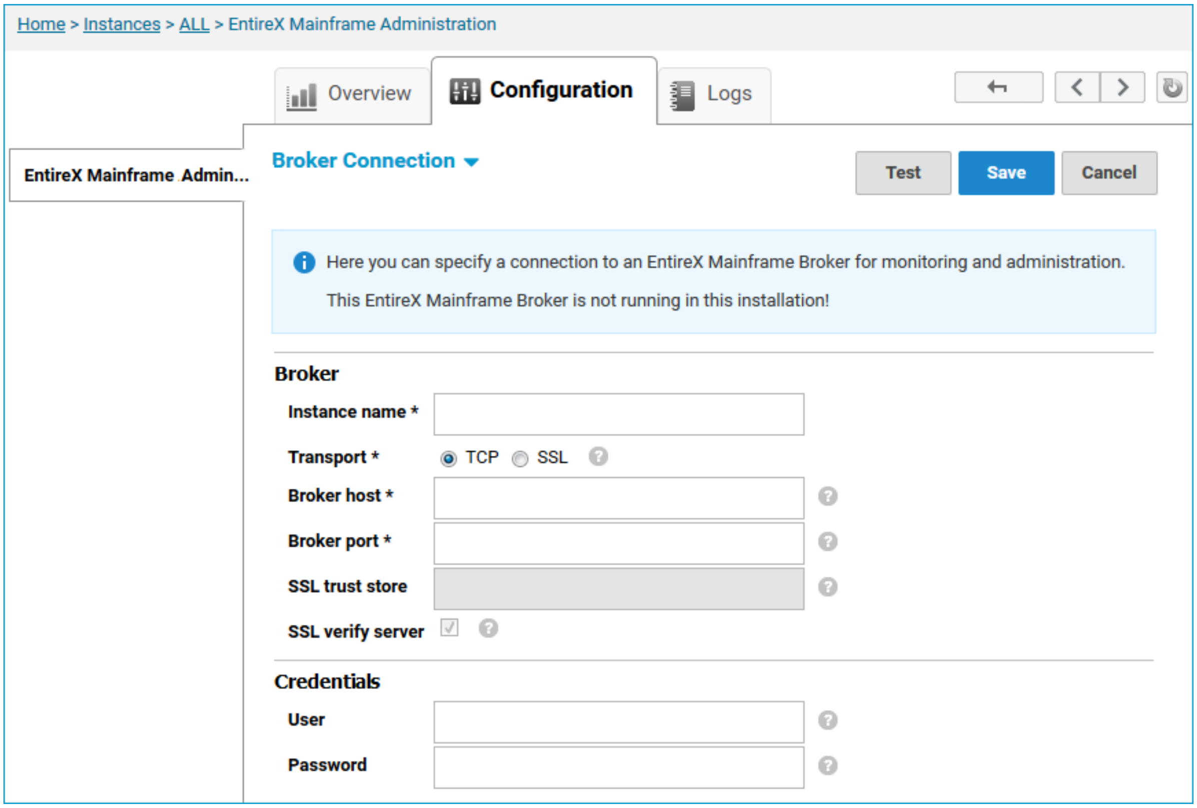


Home > Instances > ALL > EntireX Mainframe Administration

Overview
Configuration
Logs
EntireX Mainframe Administration
Broker Connection
+ −
Search for values using a text string

Instance	Host	Port

2 Click the + button in the upper right corner to add a mainframe broker connection.



Home > Instances > ALL > EntireX Mainframe Administration

Here you can specify a connection to an EntireX Mainframe Broker for monitoring and administration.

This EntireX Mainframe Broker is not running in this installation!

Broker

Instance name *

Transport * ● TCP ○ SSL

Broker host *

Broker port *

SSL trust store

SSL verify server ☑

Credentials

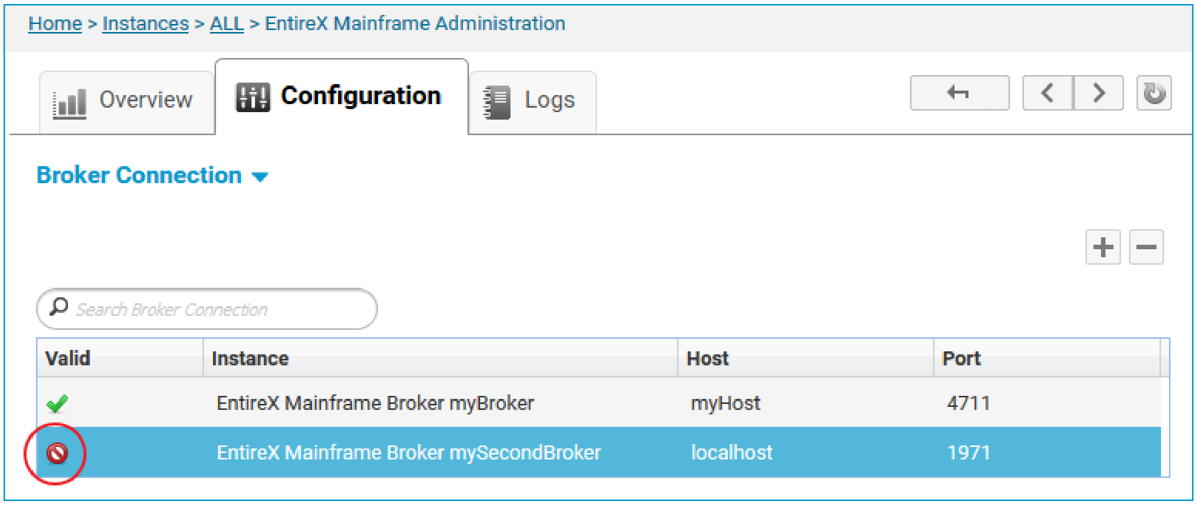
User

Password

Parameter	Description
Broker	
Instance name	Required. Provide unique instance name. Permitted characters: letters, numbers, hyphen (-), underscore (_) and dot (.).
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.

3 Click **Edit** to configure the broker connection.

4 Click **Test** to check the correctness of your input, or **Save** to apply your changes.



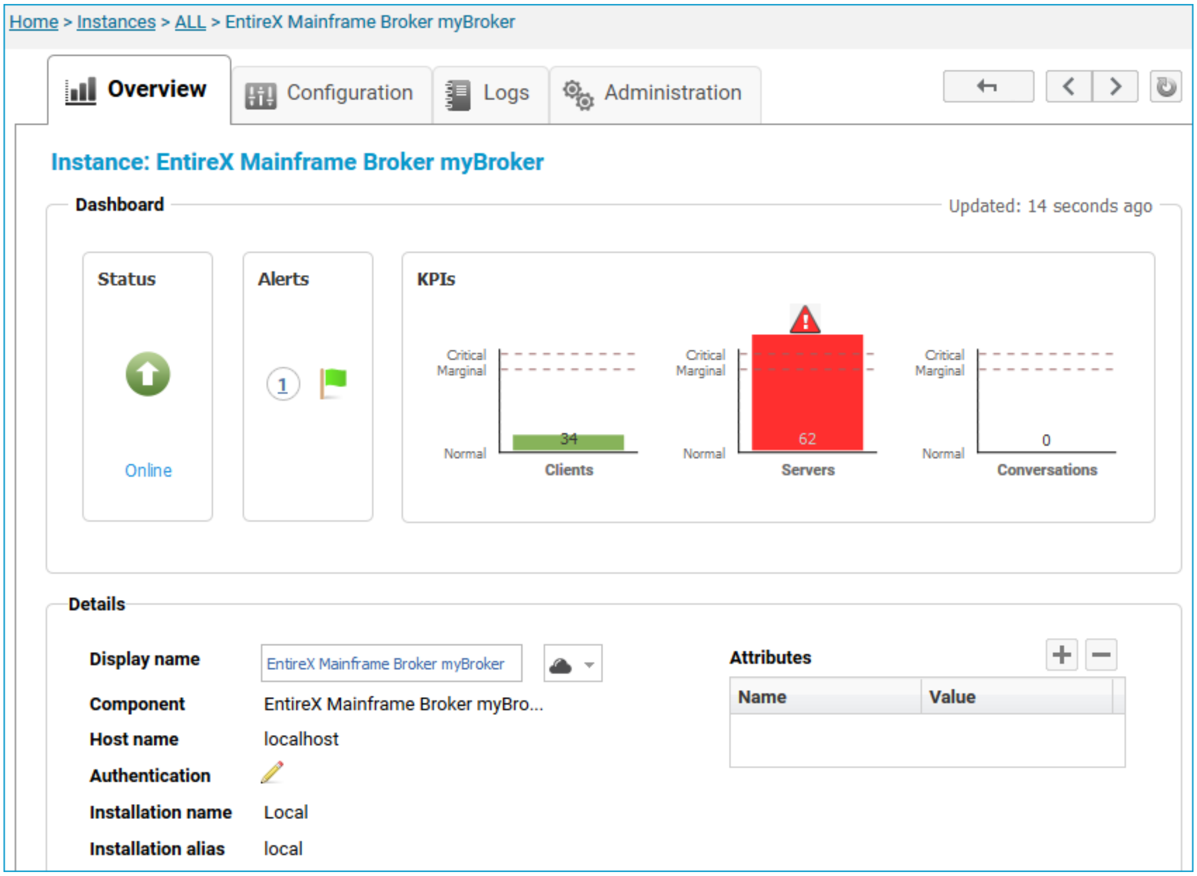
Valid	Instance	Host	Port
✓	EntireX Mainframe Broker myBroker	myHost	4711
⊘	EntireX Mainframe Broker mySecondBroker	localhost	1971

If the instance is not valid, click the **Logs** tab of the instance for more information in the *connection.log* file.

Viewing the Runtime Status

To view the runtime status of the EntireX Mainframe Broker

- Navigate to **Home > Instances > ALL > EntireX Mainframe Broker <**`instance name`**>** and click the **Overview** tab.


Home > Instances > ALL > EntireX Mainframe Broker myBroker

Instance: EntireX Mainframe Broker myBroker

Dashboard — Updated: 14 seconds ago

Details

Display name: EntireX Mainframe Broker myBroker
Component: EntireX Mainframe Broker myBro...
Host name: localhost
Authentication
Installation name: Local
Installation alias: local

Attributes

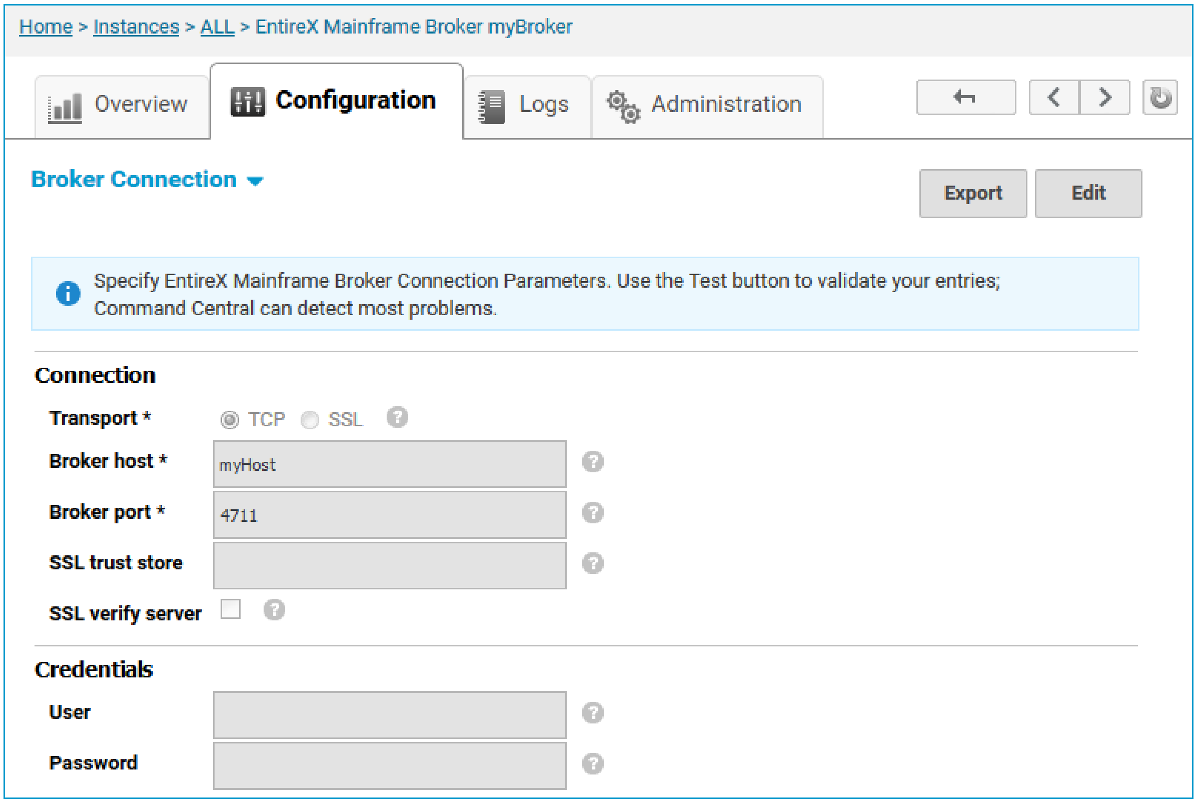
Name	Value

Status	Description
Unresponsive	The EntireX Mainframe Broker has not answered yet; the status is shown as unresponsive. This is the default status after creating an EntireX Mainframe Broker connection. For more information click the **Logs** tab to see the *connection.log* file.
Stopped	The EntireX Mainframe Broker is down after successful communication.
Error	If the EntireX Mainframe Broker communication returns an error, the status is shown as error. For more information click the **Logs** tab to see the *connection.log* file.
Online	The EntireX Mainframe Broker is running.

Configuring an EntireX Mainframe Broker Connection

To configure EntireX Mainframe Broker

1 Navigate to **Home > Instances > ALL > EntireX Mainframe Broker** <*`instance name`*> and click the **Configuration** tab.



Specify EntireX Mainframe Broker Connection Parameters. Use the Test button to validate your entries; Command Central can detect most problems.

Connection

Transport *	● TCP ○ SSL
Broker host *	myHost
Broker port *	4711
SSL trust store	
SSL verify server	□

Credentials

User	
Password	

Parameter	Description
Broker	
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.

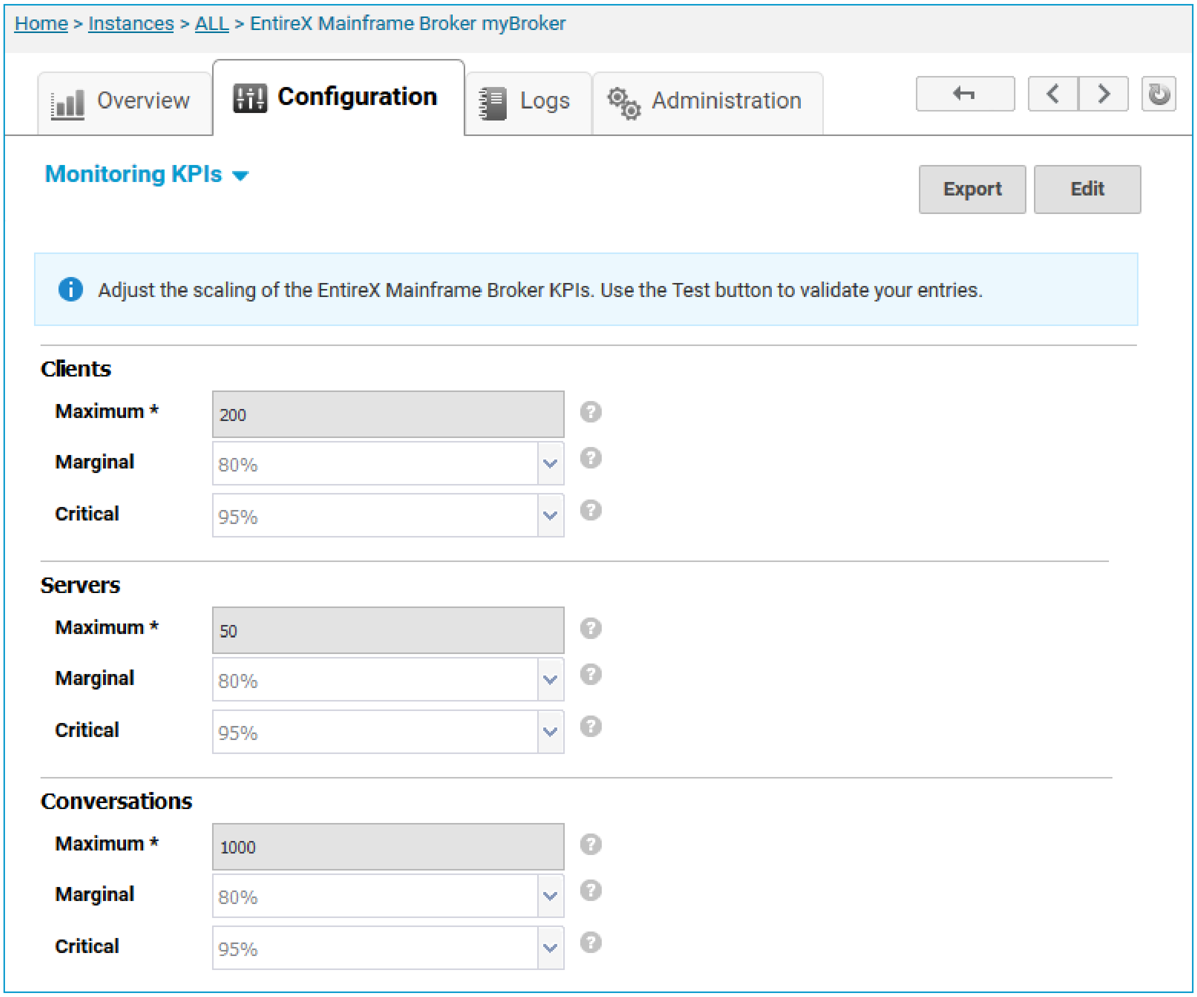
2 Click **Edit** to configure the broker connection.

3 Click **Test** to test the correctness of your input, or **Save** to apply your changes.

Configuring the Monitoring KPIs

To configure Monitoring KPIs of an EntireX Mainframe Broker

1 Navigate to **Home** > **Instances** > **ALL** > **EntireX Mainframe Broker** <*instance name*>, click the **Configuration** tab and choose **Monitoring KPIs**.



Adjust the scaling of the EntireX Mainframe Broker KPIs. Use the Test button to validate your entries.

Clients

Maximum *	200
Marginal	80%
Critical	95%

Servers

Maximum *	50
Marginal	80%
Critical	95%

Conversations

Maximum *	1000
Marginal	80%
Critical	95%

Parameter	Description
Clients	
Maximum	Maximum number of clients in the overview graph.
Marginal	Marginal barrier for numbers of clients in the overview graph.
Critical	Critical barrier for numbers of clients in the overview graph.
Servers	
Maximum	Maximum number of servers in the overview graph.
Marginal	Marginal barrier for numbers of servers in the overview graph.
Critical	Critical barrier for numbers of servers in graph.
Conversations	
Maximum	Maximum number of conversations in the overview graph.
Marginal	Marginal barrier for numbers of conversations in the overview graph.
Critical	Critical barrier for numbers of conversations in the overview graph.

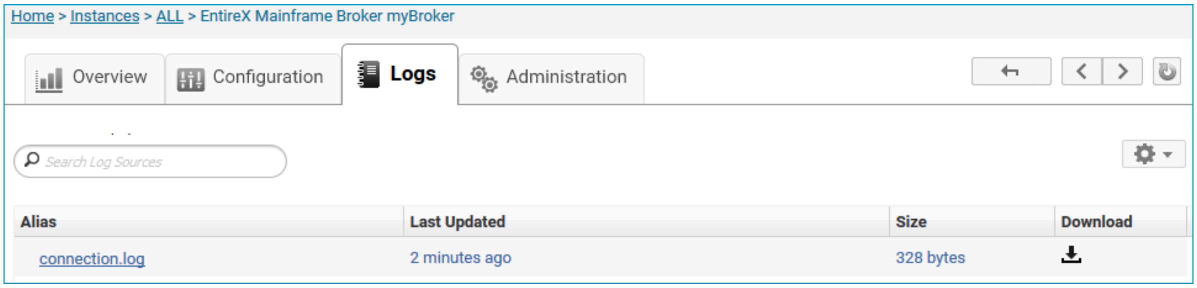
2 Click **Edit** to adjust the scaling of the EntireX Mainframe Broker KPIs.

3 Click **Test** to test the correctness of your input, or **Apply**.

Inspecting the Log Files

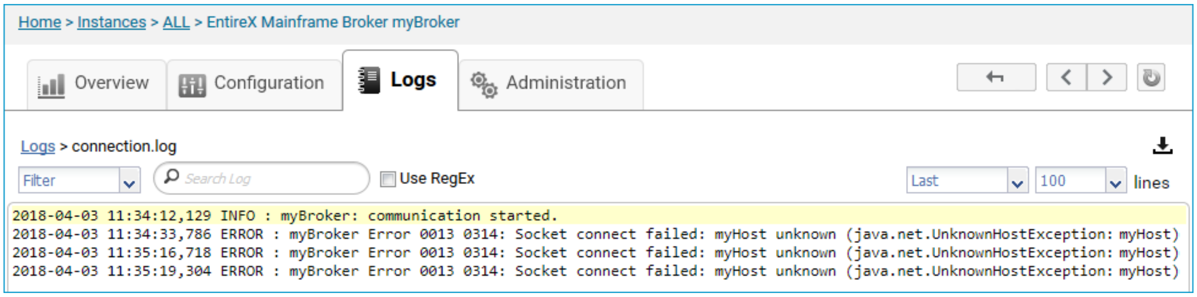
To inspect the log file of the broker connection

1 Navigate to **Home > Instances > ALL > EntireX Mainframe Broker** <*instance name*> and click the **Logs** tab.



Alias	Last Updated	Size	Download
connection.log	2 minutes ago	328 bytes	

2 In the **Alias** column you can select a log file to inspect.

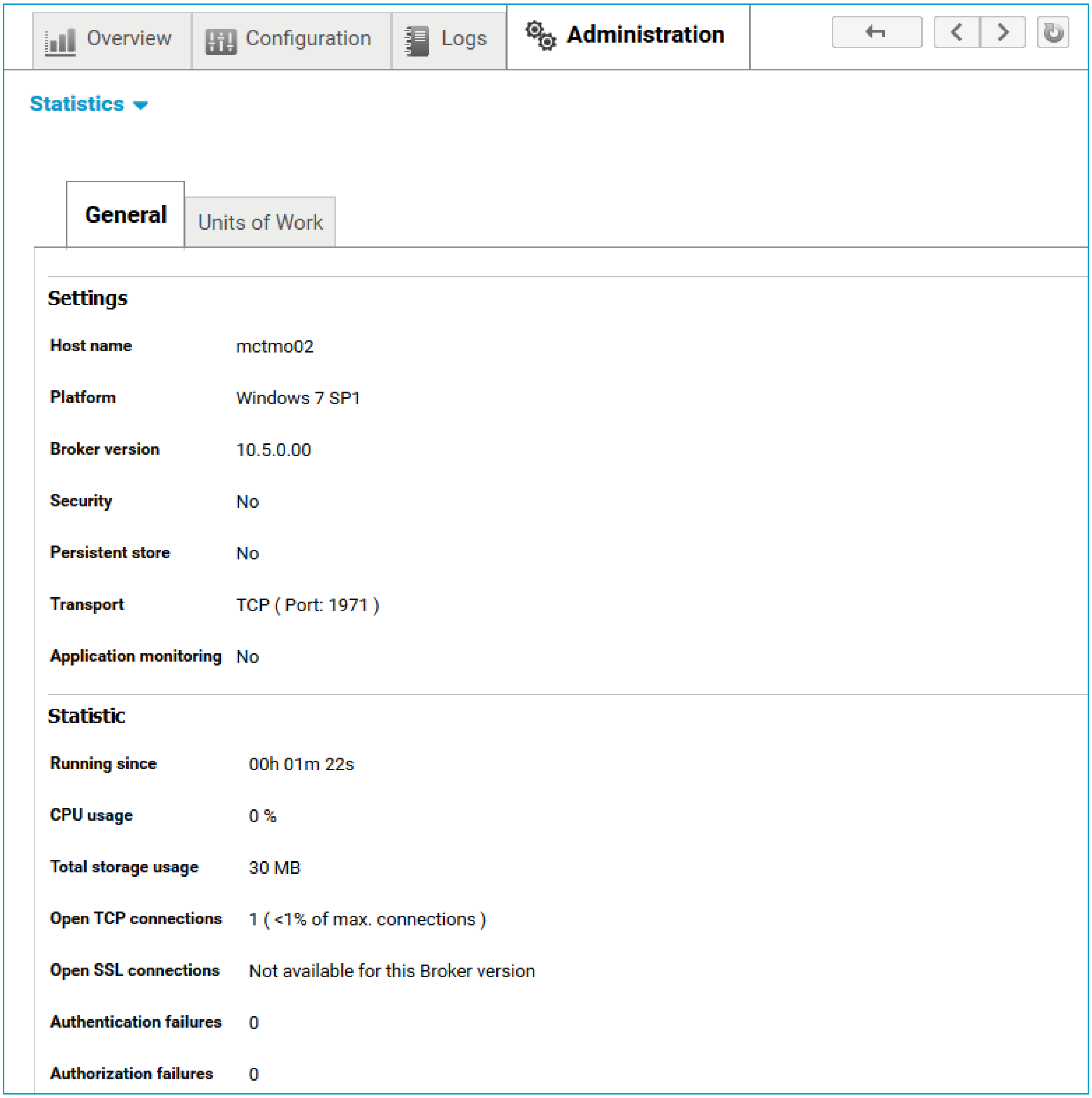


```
2018-04-03 11:34:12,129 INFO : myBroker: communication started.
2018-04-03 11:34:33,786 ERROR : myBroker Error 0013 0314: Socket connect failed: myHost unknown (java.net.UnknownHostException: myHost)
2018-04-03 11:35:16,718 ERROR : myBroker Error 0013 0314: Socket connect failed: myHost unknown (java.net.UnknownHostException: myHost)
2018-04-03 11:35:19,304 ERROR : myBroker Error 0013 0314: Socket connect failed: myHost unknown (java.net.UnknownHostException: myHost)
```

Displaying the Statistics

To display the Statistics of a running Broker instance

1. In the Command Central **Home** page, click the **Instances** tab, then click the link associated with the Broker instance for which you want to see its statistics (same as Step 1 under *Configuring an EntireX Mainframe Broker Connection*).
2. Click on the **Administration** tab
3. Choose **Statistics** in the drop-down box.
4. Choose tab **General** to see important setings and statistics, or choose tab **Units of Work** to see UOW statistics (summarized per service).

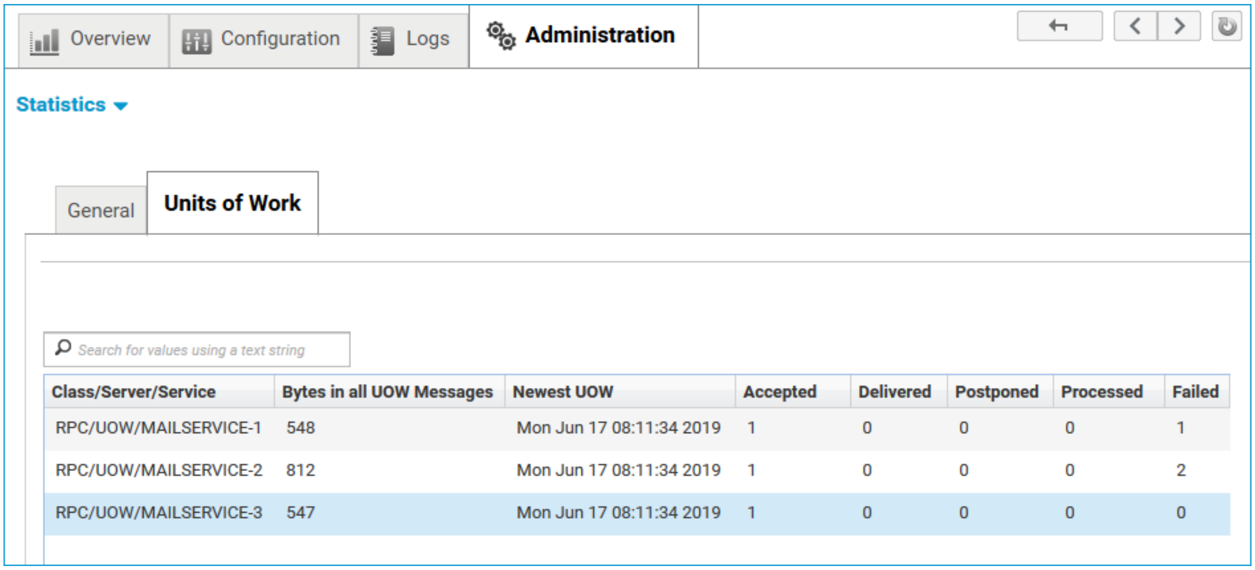


Settings

Host name	mctmo02
Platform	Windows 7 SP1
Broker version	10.5.0.00
Security	No
Persistent store	No
Transport	TCP (Port: 1971)
Application monitoring	No

Statistic

Running since	00h 01m 22s
CPU usage	0 %
Total storage usage	30 MB
Open TCP connections	1 (<1% of max. connections)
Open SSL connections	Not available for this Broker version
Authentication failures	0
Authorization failures	0



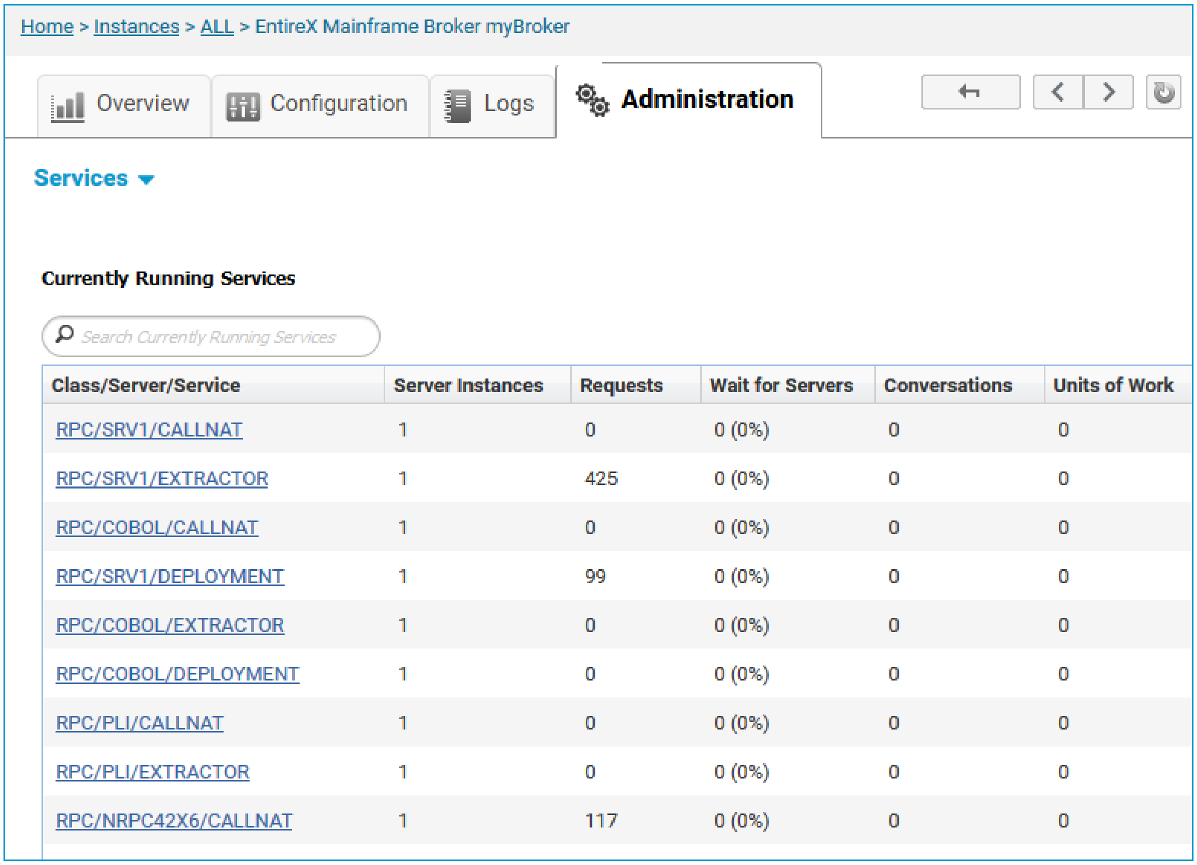
Class/Server/Service	Bytes in all UOW Messages	Newest UOW	Accepted	Delivered	Postponed	Processed	Failed
RPC/UOW/MAILSERVICE-1	548	Mon Jun 17 08:11:34 2019	1	0	0	0	1
RPC/UOW/MAILSERVICE-2	812	Mon Jun 17 08:11:34 2019	1	0	0	0	2
RPC/UOW/MAILSERVICE-3	547	Mon Jun 17 08:11:34 2019	1	0	0	0	0

Note: The returned value "UOWStatusFailed" represents the sum of UOW status "backed-out", "cancelled", "timeout" and "discarded".

Displaying Services and Servers

To view services registered to an EntireX Mainframe Broker

1 Navigate to **Home** > **Instances** > **ALL** > **EntireX Mainframe Broker** <*instance name*> and click the **Administration** tab.



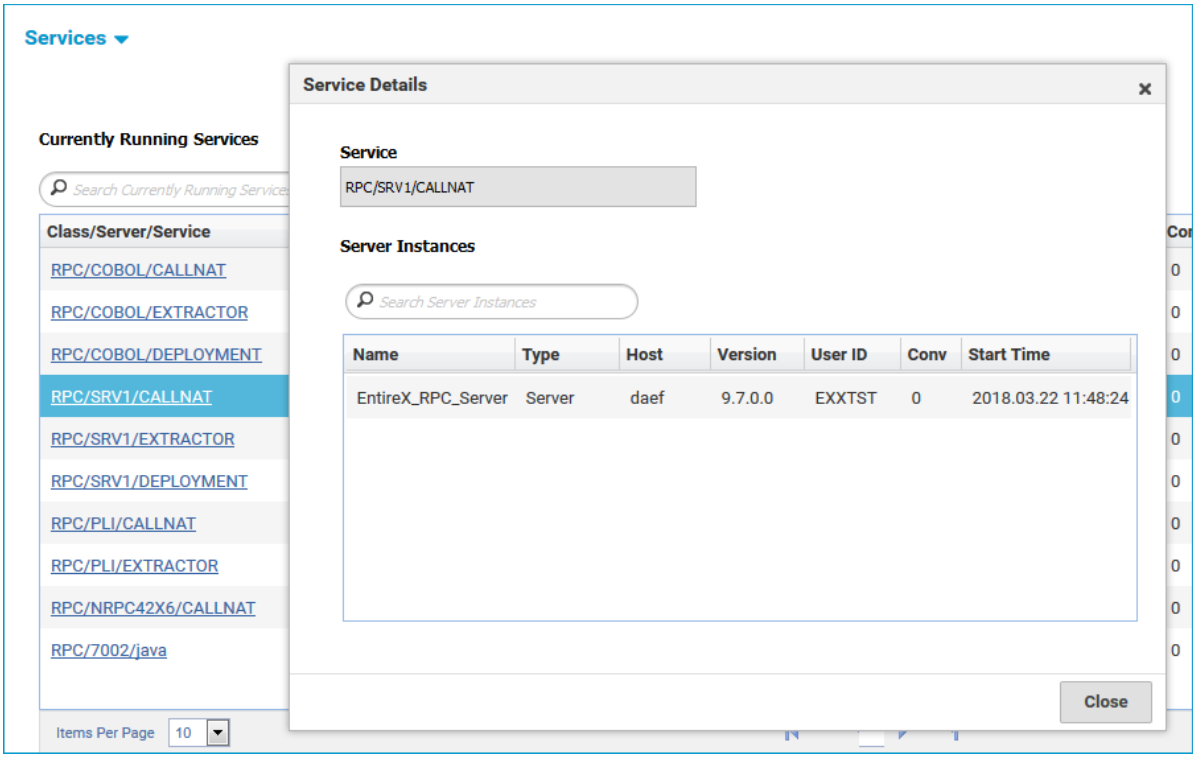
Home > Instances > ALL > EntireX Mainframe Broker myBroker
Overview
Configuration
Logs
Administration
Services

Currently Running Services

Search Currently Running Services

Class/Server/Service	Server Instances	Requests	Wait for Servers	Conversations	Units of Work
RPC/SRV1/CALLNAT	1	0	0 (0%)	0	0
RPC/SRV1/EXTRACTOR	1	425	0 (0%)	0	0
RPC/COBOL/CALLNAT	1	0	0 (0%)	0	0
RPC/SRV1/DEPLOYMENT	1	99	0 (0%)	0	0
RPC/COBOL/EXTRACTOR	1	0	0 (0%)	0	0
RPC/COBOL/DEPLOYMENT	1	0	0 (0%)	0	0
RPC/PLI/CALLNAT	1	0	0 (0%)	0	0
RPC/PLI/EXTRACTOR	1	0	0 (0%)	0	0
RPC/NRPC42X6/CALLNAT	1	117	0 (0%)	0	0

2 In the **Class/Server/Service** column, click on a service to display the servers providing this service.



Class/Server/Service	Co
RPC/COBOL/CALLNAT	0
RPC/COBOL/EXTRACTOR	0
RPC/COBOL/DEPLOYMENT	0
RPC/SRV1/CALLNAT	0
RPC/SRV1/EXTRACTOR	0
RPC/SRV1/DEPLOYMENT	0
RPC/PLI/CALLNAT	0
RPC/PLI/EXTRACTOR	0
RPC/NRPC42X6/CALLNAT	0
RPC/7002/java	0

Service Details

Service: RPC/SRV1/CALLNAT

Server Instances

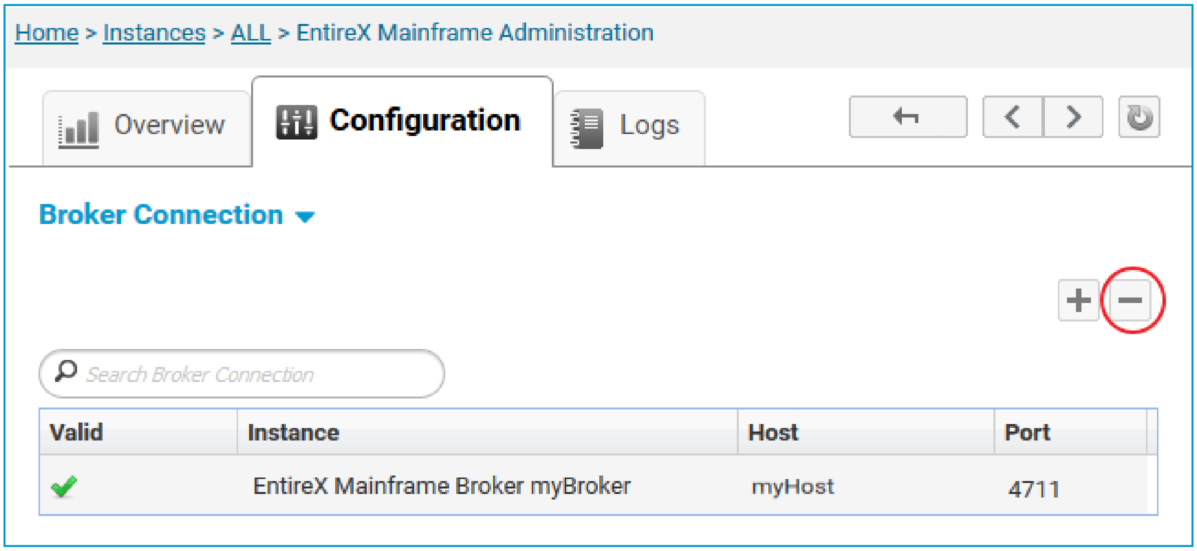
Name	Type	Host	Version	User ID	Conv	Start Time
EntireX_RPC_Server	Server	daef	9.7.0.0	EXXTST	0	2018.03.22 11:48:24

The **Start Time** is displayed in the local time where the SPM is running.

Deleting an EntireX Mainframe Broker Connection

To delete an EntireX Mainframe Broker connection

1 Navigate to **Home > Instances > ALL > EntireX Mainframe Administration** and click the **Configuration** tab.



Home > Instances > ALL > EntireX Mainframe Administration

Valid	Instance	Host	Port
✓	EntireX Mainframe Broker myBroker	myHost	4711

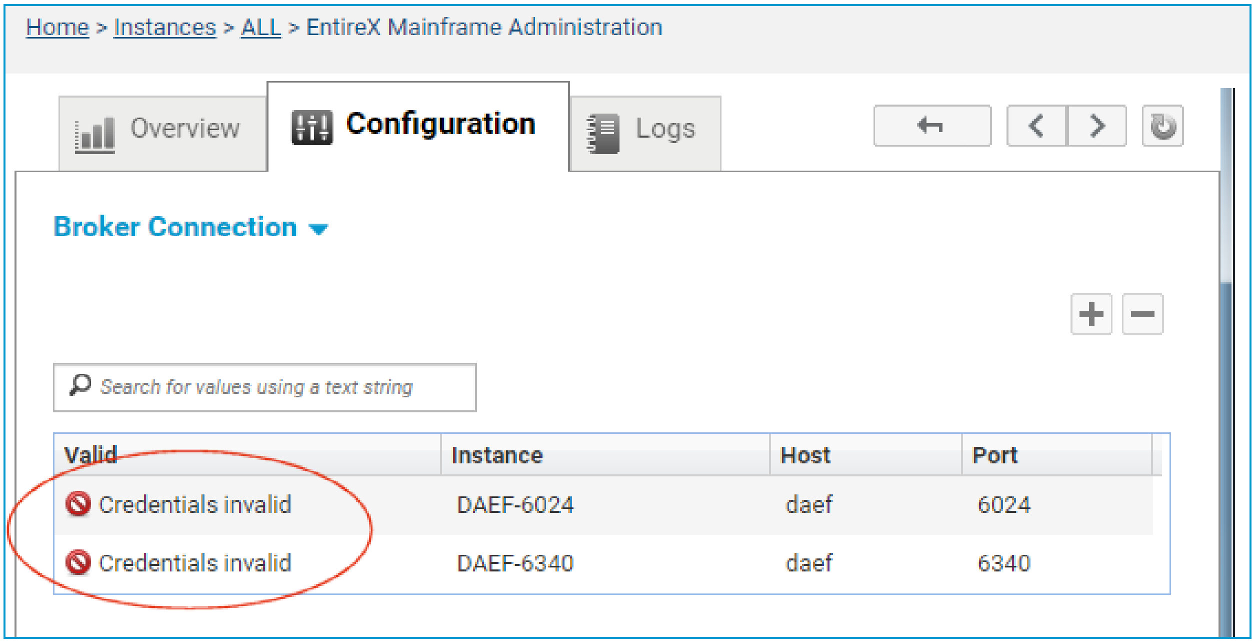
2 Select the broker connection you want to delete and press the − button in the upper right corner.

3 Click **OK** to confirm deletion of this broker connection.

Security Considerations

If you change the credentials of a secured Mainframe Broker using a security system where the number of login attempts is limited (for example RACF), this might result in your user ID being revoked. Command Central regularly tries to connect to the Mainframe Brokers to retrieve and display the latest information. If the number of rejected login attempts is reached before the Mainframe Broker Connection has been reconfigured in Command Central, the security system might block this user and the affected Mainframe Brokers cannot be accessed.

This is avoided by the following behavior: if a Mainframe Broker Connection retrieves an EntireX ACI Security Error (message class 0008) all Mainframe Broker Connections with the same user are disabled by setting their runtime status to Error preventing any more connection attempts. The affected Mainframe Broker Connections are marked by "Credentials invalid" on the **Configuration** tab of **Home > Instances > ALL > EntireX Mainframe Administration**:


Home > Instances > ALL > EntireX Mainframe Administration
Overview
Configuration
Logs
Broker Connection
+
−
Search for values using a text string

Valid	Instance	Host	Port
⊘ Credentials invalid	DAEF-6024	daef	6024
⊘ Credentials invalid	DAEF-6340	daef	6340

The connection.log of the affected Mainframe Broker Connections contains the following entry:

```
<time stamp> <instance name> Security Error occurred for a Mainframe Broker proxy ↩
with same user ID '<user ID>' => this instance is set to runtime status Error!
```

To free Mainframe Broker Connections blocked due to changed credentials

- Under **Home > Instances > ALL > EntireX Mainframe Broker** <*instance name*>, click the **Configuration** tab of the affected Mainframe Broker Connections and adjust the credentials accordingly.

  The actual runtime status is displayed, in contrast to the runtime status `Error` (see above).

Note: Because the same security system such as RACF might be used on multiple hosts, only the *user* of each Mainframe Broker Connection is considered, not the *host*. This could result in Mainframe Broker Connections being disabled even if they are not directly affected by a credentials change.

17 Monitoring EntireX Mainframe Broker using the Command Central Command Line

See also *Monitoring EntireX Mainframe Broker* and *Monitoring EntireX Mainframe Broker using the Command Central GUI*.

Creating an EntireX Mainframe Broker Connection

Parameter	Value	Description
`Instance`	*`instance`*	Required. Name of the runtime component, for example "myBroker".
`Transport`	`TCP | SSL`	Transport over TCP or SSL. Default is `TCP`.
`Host`	*`name`*	Required. EntireX Broker host name or IP address.
`Port`	`1025-65535`	Required. Port number in range from 1025 to 65535.
`SslTrustStore`	*`filename`*	Optional. Specifies the location of the SSL trust store.
`SslVerifyServer`	`true | false`	Optional. The RPC server as SSL client checks the identity of the broker as SSL server. Default is `true`.
`User`	*`user`*	Optional. The user ID for secured access to the broker.
`Password`	*`password`*	Optional. The password for secured access to the broker.

Command	Parameter	Description
`sagcc create configuration data`	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. Must be `EntireXMainframeProxy-Administration`.
	`instanceid`	Required. Must be `EXX-BROKER`.
	`-i` *`file`*	Required. Specifies the file from where you want the input read.

Example

- To create a new instance of "EntireX Mainframe Broker", with the name "MyBroker" in the installation with alias name "local" from the file *MyBroker.json* in the current working directory:

```
sagcc create configuration data local EntireXMainframeProxy-Administration ↩
EXX-BROKER -i MyBroker.json
```

MyBroker.json

```
{
    "Instance": "MyBroker",
    "Transport": "TCP",
    "Host": "mainframeHost",
    "Port": "4713",
    "SslTrustStore": "",
    "SslVerifyServer": "false",
    "User": "",
    "Password": ""
}
```

Displaying the EntireX Mainframe Broker Connection

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 broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXMainframeProxy-Broker-"`.

Example

- To list EntireX Mainframe Broker Connection components in the installation with alias name "local":

```
sagcc list inventory components local EntireXMainframeProxy-Broker-*
```

A list of all EntireX Mainframe Broker Connection components will be displayed. If the component is not valid, you will find more information in the connection.log file.

Viewing the Runtime Status

The following table lists the parameters to include when displaying the state of an EntireX Mainframe Broker 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 broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXMainframeProxy-Broker-"`.

Example

- To display state information about the EntireX Mainframe Broker:

  ```
  sagcc get monitoring state local EntireXMainframeProxy-Broker-MyBroker
  ```

  Runtime status and runtime state will be displayed.

  - Runtime *status* indicates whether a runtime component is running, unknown or down. Examples of a runtime status are `UNRESPONSIVE`, `ONLINE`, `ERROR`, or `STOPPED`. If the EntireX Mainframe Broker is detected as a non-mainframe broker, the status is shown as `ERROR`.
  - 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.

Configuring the EntireX Mainframe Broker

Configuring the Broker Connection

The following table lists the parameters to include when updating a Broker Connection of an EntireX Mainframe Broker instance, using the Command Central `update configuration` commands.

Parameter	Value	Description
`Transport`	`TCP` | `SSL`	Transport over TCP or SSL. Default is `TCP`.
`Host`	*`name`*	Required. EntireX Broker host name or IP address.
`Port`	`1025-65535`	Required. Port number in range from 1025 to 65535.
`SslTrustStore`	*`filename`*	Optional. Specifies the location of the SSL trust store.
`SslVerifyServer`	`true` | `false`	Optional. The RPC server as SSL client checks the identity of the broker as SSL server. Default is `true`.
`User`	*`user`*	Optional. The user ID for secured access to the broker.
`Password`	*`password`*	Optional. The password for secured access to the broker.

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

Example

- To update an instance of "EntireX Mainframe Broker Connection", with the name "MyBroker" in the installation with alias name "local" from the file *MyBroker.json* in the current working directory:

```
sagcc update configuration data local EntireXMainframeProxy-Administration ↩
EXX-BROKER -i MyBroker.json
```

MyBroker.json

```
{
    "Transport": "TCP",
    "Host": "mainframeHost",
    "Port": "9999",
    "SslTrustStore": "",
    "SslVerifyServer": "false",
    "User": "",
    "Password": ""
}
```

Configuring the Monitoring KPIs

The following table lists the parameters to include when updating the Monitoring KPIs of an EntireX Mainframe Broker instance, using the Command Central `update configuration` commands.

Parameter	Value	Description
`ClientsMaximum`	1-2147483647	Required. Maximum number of clients in graph.
`ClientsCriticalPercent`	1-100	Required. Critical barrier of clients in graph in %.
`ClientsMarginalPercent`	1-100	Required. Marginal barrier of clients in graph in %.
`ServersMaximum`	1-2147483647	Required. Maximum number of servers in graph.
`ServersCriticalPercent`	1-100	Required. Critical barrier of servers in graph in %.
`ServersMarginalPercent`	1-100	Required. Marginal barrier of servers in graph in %.
`ConversationsMaximum`	1-2147483647	Required. Maximum number of conversations in graph.
`ConversationsCriticalPercent`	1-100	Required. Critical barrier of conversations in graph in %.
`ConversationsMarginalPercent`	1-100	Required. Marginal barrier of conversations in graph in %.

Command	Parameter	Description
`sagcc update configuration data`	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXMainframeProxy-Broker-`".
	`instanceid`	Required. Must be `EXX-MONITORING-KPIS`.
	`-i` *`file`*	Required. Specifies the file from where you want the input read.

Example

- To update an instance of "EntireX Mainframe Broker", with the name "MyBroker" in the installation with alias name "local" from the file MyKpis.json in the current working directory:

```
sagcc update configuration data local EntireXMainframeProxy-Broker-MyBroker ↩
EXX-MONITORING-KPIS -i MyKpis.json
```

MyKpis.json

```
{
    "ClientsMaximum": "200",
    "ClientsCriticalPercent": "95",
    "ClientsMarginalPercent": "80",
    "ServersMaximum": "50",
    "ServersCriticalPercent": "95",
    "ServersMarginalPercent": "80",
    "ConversationsMaximum": "1000",
    "ConversationsCriticalPercent": "95",
    "ConversationsMarginalPercent": "80"
}
```

Inspecting the Log Files

Here you can administer the log files of the EntireX Mainframe Broker Connection instance.

- Listing all EntireX Broker Log Files
- Getting Content from or Downloading RPC Server Log Files

Listing all EntireX Broker Log Files

The following table lists the parameters to include when displaying or modifying parameters of the EntireX Mainframe Broker, using the Command Central `list` commands.

Command	Parameter	Description
`sagcc list diagnostics logs`	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXMainframeProxy-Broker-`".

Example

- To list the log files of EntireX Mainframe Broker Connection instance, in the installation with alias name "local" on stdout:

```
sagcc list diagnostics logs local EntireXMainframeProxy-Broker-MyBroker
```

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 broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXMainframeProxy-Broker-`".
	`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.

Examples

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

```
sagcc get diagnostics logs local EntireXMainframeProxy-Broker-MyBroker ↩
connection.log tail
```

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

```
sagcc get diagnostics logs local EntireXMainframeProxy-Broker-MyBroker export -o ↩
myfile.zip
```

Displaying the Statistics

Displaying the General Statistics of a Running EntireX Broker

Here you can display the current statistics of a running EntireX Broker.

Command	Parameter	Description
`get administration`	*`component`*	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 "`EntireXMainframeProxy-Broker-`".
	`Statistics`	Required. Specifies what is to be administered.

Command	Parameter	Description
	`loadStatisticGeneral`	Required. Get the general settings.
	`-f xml|json`	Required. Specifies XML or JSON as output format.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Examples

- To display the current general setting of the running EntireX Broker with the name 'MyBroker' in the installation with alias name 'local'.broker in JSON format on stdout:

```
sagcc get administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Statistics loadStatisticGeneral -f json
```

- To display the current general setting of the running EntireX Broker with the name 'MyBroker' in the installation with alias name 'local' in XML format on stdout:

```
sagcc get administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Statistics loadStatisticGeneral -f xml
```

Displaying the UOW (Unit of Work) Statistics of a Running EntireX Broker

Here you can display the current UOW statistics of a running EntireX Broker. The UOW statistics are summarized per service.

Note: The returned value "UOWStatusFailed" represents the sum of UOW status "backed-out", "cancelled", "timeout" and "discarded".

Command	Parameter	Description
`get administration`	*`component`*	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 "`EntireXMainframeProxy-Broker-`".
	`Statistics`	Required. Specifies what is to be administered.
	`loadStatisticUow`	Required. Get the statistics of UOW usage.
	`-f xml|json`	Required. Specifies XML or JSON as output format.
	`-o` *`file`*	Optional. Specifies the file where you want the output written.

Examples

- To display the current UOW statistics of the running EntireX Broker with the name 'MyBroker' in the installation with alias name 'local'.broker in JSON format on stdout:

```
sagcc get administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Statistics loadStatisticUow -f json
```

- To display the current UOW statistics of the running EntireX Broker with the name 'MyBroker' in the installation with alias name 'local' in XML format on stdout:

```
sagcc get administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Statistics loadStatisticUow -f xml
```

Monitoring Services

List Running Services

Command	Parameter	Description
`sagcc list administration`	`component`	Specifies that a component will be administered.
	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is `"EntireXMainframeProxy-Broker-"`.
	`Services`	Required. Specifies what is to be administered.
	`listServices`	Required. List all services.
	`-f xml|json`	Required. Specifies XML or JSON as output format.

Examples

- To display a list of services of the running EntireX Mainfram Broker with the name "MyBroker" in the installation with alias name "local" in JSON format:

```
sagcc list administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Services listServices -f json
```

- To store a list of services of the EntireX Mainframe Broker with the name "MyBroker" in the installation with alias name "local" in JSON format in the file services.json of the current working directory:

```
sagcc list administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Services listServices -o services.json
```

List Server Instances

Command	Parameter	Description
`sagcc list administration`	`component`	Specifies that a component will be administered.
	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXMainframeProxy-Broker-`".
	`Services`	Required. Specifies what is to be administered.
	`listServers`	Required. List all servers.
	`serviceName`	Required. Shows only servers of this service. Format: `class/server/service`.
	`-f xml|json`	Required. Specifies XML or JSON as output format.

Examples

- To display a list of servers of the current service of the EntireX Mainframe Broker with the name "MyBroker" in the installation with alias name "local" in XML format:

```
sagcc list administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Services listServers serviceName=RPC/SRV1/CALLNAT -f xml
```

- To store a list of servers in JSON format in the file services.json of the current working directory:

```
sagcc list administration component local EntireXMainframeProxy-Broker-MyBroker ↩
Services listServers serviceName=RPC/SRV1/CALLNAT -o server.json
```

Deleting an EntireX Mainframe Broker Connection

The following table lists the parameters to include when deleting an EntireX Mainframe Broker Connection instance, using the Command Central `delete configuration` commands.

Command	Parameter	Description
`sagcc delete configuration data`	`node_alias`	Required. Specifies the alias name of the installation in which the broker connection is installed.
	`componentid`	Required. The component identifier. The prefix is "`EntireXMainframeProxy-Broker-`".

Example

- To delete an instance of an EntireX Mainframe Broker Connection with the name "MyBroker" in the installation with alias name "local":

```
sagcc delete configuration data local EntireXMainframeProxy-Administration ↩
EntireXMainframeProxy-Broker-MyBroker --force
```

18 Introduction to Administering EntireX RPC Servers using Command Central (UNIX and Windows)

See also:

- *Administering the EntireX RPC Server for* C | CICS Socket Listener | .NET | IMS Connect | Java | IBM MQ | XML/SOAP *using the Command Central GUI*
- *Administering the EntireX RPC Server for* C | CICS Socket Listener | .NET | IMS Connect | Java | IBM MQ | XML/SOAP *using the Command Central Command Line*

Note: Command Central functionality that is not EntireX-specific is described in the separate Command Central documentation or the online help provided with Command Central. On Empower, the documentation is provided under webMethods > EntireX > EntireX 10.5 > Additional Documentation.

Scope

This section applies to RPC server instances running on UNIX and Windows platforms. You can use Command Central to perform the following operations on EntireX RPC servers:

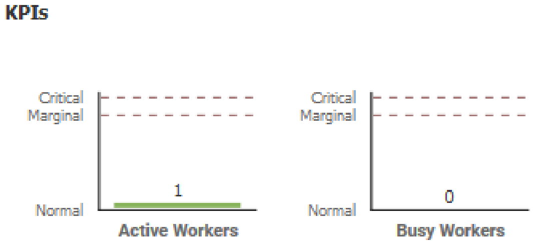
- View the EntireX RPC servers running in each environment of your IT landscape
- View the versions of EntireX RPC servers
- Monitor EntireX RPC server installations
- Monitor runtime status, KPIs (key performance indicators), and alerts of EntireX RPC server instances
- Start, stop, and restart EntireX RPC servers
- Configure the following parameters of EntireX RPC servers:
  - Broker connection parameters
  - Configuration file
  - License keys
  - Monitoring KPIs
  - Server settings
  - Trace
  - Classpath *
  - Library locations *
  - CICS connection parameters *
  - MQ connection parameters *
  - IMS connection parameters *
  - NET Library parameters *

  - XML deployment parameters *
  - XML mapping file parameters *
- Enable and specify EntireX RPC trace level
- Create new EntireX RPC servers
- Delete existing EntireX RPC servers

Note: Configuration parameters marked with an asterisk (*) do not apply to all types of RPC server (see *Supported Configuration Types* below). These parameters are described in the relevant sections.

Monitoring EntireX RPC Server KPIs

The visual key performance indicators (KPIs) and alerts enable you to monitor a webMethods EntireX RPC Server's health. The following KPIs help you administer, troubleshoot, and resolve performance issues in EntireX RPC servers:



KPI	Description
Active Workers	Absolute number of active workers.
Busy Workers	Absolute number of busy servers.

Supported Configuration Types

The EntireX RPC server component supports the configuration instances listed in the following table.

Instance	Type	Use to...	Applies to
BROKER	BROKER	Configure broker connection settings	All
CONFIGURATION-FILE	CONFIGURATION-FILE	Show and edit the RPC server configuration file	All
LICENSE-KEYS	LICENSE-KEYS	Show and set the license key file	All
MONITORING-KPI	MONITORING-KPI	Show and edit the monitoring KPI settings, such as marginal and critical bounds, etc.	All
SERVER	SERVER	Configure server settings	All
TRACE	TRACE	Show and edit the EntireX RPC server trace level	All
CLASSPATH	CLASSPATH	Classpath to the RPC server implementation	Java
LIBRARY-LOCATIONS	LIBRARY-LOCATIONS	Path to the library containing the C server programs	C
CICS	CICS	CICS-specific parameters	CICS Socket Listener
MQ	MQ	MQ-specific parametrers	IBM MQ
IMS	IMS	IMS-specific parameters	IMS Connect
.NET	.NET	.NET-specific parameters	.NET
XML MAPPING FILES	XML MAPPING FILES	The list of XML mapping files configured for this RPC server	XML/SOAP
DEPLOYMENT	DEPLOYMENT	Allow dynamic deployment of XML mapping files.	