webMethods EntireX

Administration

Version 9.5 SP1

November 2013

This document applies to webMethods EntireX Version 9.5 SP1.

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

Document ID: EXX-ADMIN-95SP1-20140628GENERAL

Table of Contents

1 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	Win	UNIX	z/VM			
SAG			x		R	Root directory for all Software AG infrastructure products (e.g. System Management Hub, Software AG Web Server).	
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. Not required by EntireX Mini Runtime.	See *Shell Environment Settings* under *Post-installation Steps under UNIX*.
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* under *Post-installation Steps under UNIX*.
SHLIB_PATH			x		R	Same as LD_LIBRARY_PATH on HP-UX.	See *Shell Environment Settings* under *Post-installation Steps under UNIX*.
LIBPATH			x		R	Same as LD_LIBRARY_PATH on AIX.	See *Shell Environment Settings* under *Post-installation Steps under UNIX*.
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)	

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Win	UNIX	z/VM			
						or during installation (Windows).	
ARGDIR			x		R	Home directory of the System Management Hub	See *System Management Hub for EntireX*.
ARGVERS			x		R	Version of the System Management Hub	
ETB_ATTR		x	x		O	Value of Broker attribute file. Set automatically by the Broker startup shell script.	See *Broker Attributes* in the administration documentation.
ETB_LOG		x	x		O	Accounting file.	See *Accounting in EntireX Broker* in the general administration documentation.
ETB_NONACT	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.
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 *Administration of Broker Stubs* in the platform-specific administration documentation.
ETB_STUBLOG	x	x	x	x	O	Trace level for the EntireX Broker API.	See *Application Stublog File* in the UNIX administration documentation | *Tracing for Stubs* under z/OS | | z/VM.
ETB_STUBLOGPATH		x	x		O	Under UNIX and Windows, the directory where the log file is created if ETB_STUBLOG is used.	
ETB_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*

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Win	UNIX	z/VM			
							webMethods EntireX in the platform-specific administration documentation.
ETB_TRANSPORT	x	x	x		O	Sets the default transport method for Broker stubs.	See *Setting 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* under *Post-installation Steps under UNIX*.
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 Broker, it might be necessary to set a preferred API Version for the Broker.
ERX_CODEPAGE		x	x		O	Sets the locale string to be used for internationalization with the EntireX RPC Runtime.	Internationalization for various EntireX components such as DCOM Wrapper, .NET Wrapper and C Wrapper, if communicating with EntireX Broker version 7.1.x and below. See *Preparing EntireX Components for Internationalization*.

Environment Variable	Platform				Opt/ Req	Description	More Information
	z/OS	Win	UNIX	z/VM			
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.
NA2_BKDIAG		x	x		O	Security exit diagnostics. Use only if requested by Software AG support.	
NA2_BKPRIV		x	x	x	O	Security exit setting.	See *Setting up EntireX Security for Broker Kernel* in the UNIX and Windows post-installation documentation; *Step 4: Rename SECUEXI0 to SECUEXIT for Security (Optional)* in the z/VM installation documentation.
REGFILE			x		R	RGS repository for Software AG Base Technology components under UNIX.	

Using Environment Variables under z/OS

In Batch, CICS and IMS, use the SAGTOKEN Utility to set and delete environment variables. See *SAGTOKEN Utility* under *Administration of Broker Stubs under z/OS* in the z/OS administration documentation.

In Com-plete, use the EXAENV environment store to set and delete environment variables. See *EXAENV Environment Store* under *Administration of Broker Stubs under z/OS*.

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/OSD (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/OSD.

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

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`

2 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 */opt/softwareag/EntireX/config/etb/<brokerid>*.

Example: */opt/softwareag/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 */opt/softwareag/EntireX/security_exit*.

Windows

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

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 */opt/softwareag/EntireX/etc*.

Windows

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

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

3 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 conversions 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-PUBLICATION`
- `NUM-PUBLISHER`
- `NUM-SERVER`
- `NUM-SERVICE`
- `NUM-SERVICE-EXTENSION`
- `NUM-SHORT[-BUFFER]`
- `NUM-SUBSCRIBER`
- `NUM-SUBSCRIBER-TOTAL`
- `NUM-TOPIC`
- `NUM-TOPIC-EXTENSION`
- `NUM-TOPIC-TOTAL`
- `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 stabile 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 re-starting 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.

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/OSD

`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 ↩
12:28:58.768  Allocated
HEAP POOL                 0x25EB4010    1050692 bytes    1457876 bytes  2009-06-26 ↩
12:28:58.769  Allocated
COMMUNICATION POOL        0x25FB5010   16781380 bytes   18239256 bytes  2009-06-26 ↩
12:28:58.769  Allocated
ACCOUNTING POOL           0x26FB7010     762052 bytes   19001308 bytes  2009-06-26 ↩
12:28:58.769  Allocated
BROKER POOL               0x27072010      61540 bytes   19062848 bytes  2009-06-26 ↩
12:28:58.775  Allocated
CONVERSATION POOL         0x27082010     368964 bytes   19431812 bytes  2009-06-26 ↩
12:28:58.775  Allocated
CONNECTION POOL           0x270DD010     233668 bytes   19665480 bytes  2009-06-26 ↩
12:28:58.779  Allocated
LONG MESSAGES POOL        0x27117010    4395204 bytes   24060684 bytes  2009-06-26 ↩
12:28:58.782  Allocated
SHORT MESSAGES POOL       0x27549010    3703876 bytes   27764560 bytes  2009-06-26 ↩
```

```
12:28:58.806  Allocated
PARTICIPANT POOL          0x278D2010    134244 bytes   27898804 bytes  2009-06-26 ↩
12:28:58.827  Allocated
PARTICIPANT EXTENSION POOL  0x278F3010     36996 bytes   27935800 bytes  2009-06-26 ↩
12:28:58.829  Allocated
PROXY QUEUE POOL          0x278FD010     26724 bytes   27962524 bytes  2009-06-26 ↩
12:28:58.829  Allocated
SERVICE ATTRIBUTES POOL    0x27904010    131668 bytes   28094192 bytes  2009-06-26 ↩
12:28:58.829  Allocated
SERVICE POOL              0x27925010     54372 bytes   28148564 bytes  2009-06-26 ↩
12:28:58.830  Allocated
SERVICE EXTENSION POOL     0x27933010     32900 bytes   28181464 bytes  2009-06-26 ↩
12:28:58.831  Allocated
TIMEOUT QUEUE POOL         0x2793C010     87268 bytes   28268732 bytes  2009-06-26 ↩
12:28:58.831  Allocated
TRANSLATION POOL          0x27952010    179300 bytes   28448032 bytes  2009-06-26 ↩
12:28:58.832  Allocated
UNIT OF WORK POOL         0x2797E010    176324 bytes   28624356 bytes  2009-06-26 ↩
12:28:58.834  Allocated
WORK QUEUE POOL           0x279AA010    391268 bytes   29015624 bytes  2009-06-26 ↩
12:28:58.835  Allocated
BLACKLIST POOL            0x27A0A010     42084 bytes   29057708 bytes  2009-06-26 ↩
12:28:58.838  Allocated
SUBSCRIPTION POOL         0x27A15010    344148 bytes   29401856 bytes  2009-06-26 ↩
12:28:58.839  Allocated
TOPIC ATTRIBUTES POOL      0x27A6A010    129620 bytes   29531476 bytes  2009-06-26 ↩
12:28:58.841  Allocated
TOPIC POOL                0x26FB6068      2952 bytes   29534428 bytes  2009-06-26 ↩
12:28:58.842  Allocated
TOPIC EXTENSION POOL       0x27A8A010     30852 bytes   29565280 bytes  2009-06-26 ↩
12:28:58.842  Allocated
PSTORE SUBSCRIBER POOL     0x27A92010     33892 bytes   29599172 bytes  2009-06-26 ↩
12:28:58.843  Allocated
PSTORE TOPIC POOL         0x27A9B010     19540 bytes   29618712 bytes  2009-06-26 ↩
12:28:58.843  Allocated
COMMUNICATION POOL         0x25FB5010   16781380 bytes   12837332 bytes  2009-06-26 ↩
12:30:58.514  Deallocated
ACCOUNTING POOL           0x26FB7010    762052 bytes   12075280 bytes  2009-06-26 ↩
12:30:58.515  Deallocated
BROKER POOL               0x27072010     61540 bytes   12013740 bytes  2009-06-26 ↩
12:30:58.516  Deallocated
CONVERSATION POOL          0x27082010    368964 bytes   11644776 bytes  2009-06-26 ↩
12:30:58.518  Deallocated
CONNECTION POOL           0x270DD010    233668 bytes   11411108 bytes  2009-06-26 ↩
12:30:58.519  Deallocated
LONG MESSAGES POOL         0x27117010   4395204 bytes    7015904 bytes  2009-06-26 ↩
12:30:58.520  Deallocated
SHORT MESSAGES POOL        0x27549010   3703876 bytes    3312028 bytes  2009-06-26 ↩
12:30:58.526  Deallocated
PROXY QUEUE POOL          0x278FD010     26724 bytes    3285304 bytes  2009-06-26 ↩
12:30:58.530  Deallocated
SUBSCRIPTION POOL         0x27A15010    344148 bytes    2941156 bytes  2009-06-26 ↩
```

```
12:30:58.530  Deallocated
TOPIC ATTRIBUTES POOL      0x27A6A010    129620 bytes    2811536 bytes  2009-06-26 ↩
12:30:58.531  Deallocated
TOPIC POOL                0x26FB6068      2952 bytes    2808584 bytes  2009-06-26 ↩
12:30:58.531  Deallocated
TOPIC EXTENSION POOL       0x27A8A010     30852 bytes    2777732 bytes  2009-06-26 ↩
12:30:58.531  Deallocated
TIMEOUT QUEUE POOL         0x2793C010     87268 bytes    2690464 bytes  2009-06-26 ↩
12:30:58.532  Deallocated
UNIT OF WORK POOL          0x2797E010    176324 bytes    2514140 bytes  2009-06-26 ↩
12:30:58.533  Deallocated
WORK QUEUE POOL            0x279AA010    391268 bytes    2122872 bytes  2009-06-26 ↩
12:30:58.533  Deallocated
BLACKLIST POOL             0x27A0A010     42084 bytes    2080788 bytes  2009-06-26 ↩
12:30:58.534  Deallocated
PSTORE SUBSCRIBER POOL      0x27A92010     33892 bytes    2046896 bytes  2009-06-26 ↩
12:30:58.534  Deallocated
PSTORE TOPIC POOL          0x27A9B010     19540 bytes    2027356 bytes  2009-06-26 ↩
12:30:58.534  Deallocated
PARTICIPANT POOL           0x278D2010    134244 bytes    1893112 bytes  2009-06-26 ↩
12:49:25.817  Deallocated
PARTICIPANT EXTENSION POOL  0x278F3010     36996 bytes    1856116 bytes  2009-06-26 ↩
12:49:25.818  Deallocated
SERVICE ATTRIBUTES POOL     0x27904010    131668 bytes    1724448 bytes  2009-06-26 ↩
12:49:25.818  Deallocated
SERVICE POOL               0x27925010     54372 bytes    1670076 bytes  2009-06-26 ↩
12:49:25.818  Deallocated
SERVICE EXTENSION POOL      0x27933010     32900 bytes    1637176 bytes  2009-06-26 ↩
12:49:25.819  Deallocated
TRANSLATION POOL           0x27952010    179300 bytes    1457876 bytes  2009-06-26 ↩
12:49:25.819  Deallocated
HEAP POOL                 0x25EB4010   1050692 bytes     407184 bytes  2009-06-26 ↩
12:49:25.820  Deallocated
KERNEL POOL                0x25E48010     407184 bytes          0 bytes  2009-06-26 ↩
12:49:25.820  Deallocated
```

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 maximum number of TCP/IP connections per communicator:

Platform	Maximum Number of TCP/IP Connections per Communicator
AIX	2,048
BS2000/OSD	2,048
HP-UX	2,048
Linux	4,096
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 and UNIX. See `POLL`.

See also `MAX-CONNECTIONS` under *`TCP-OBJECT (Struct INFO_TCP)`* under *Information Reply Structures* in the Broker CIS 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 UNIX

Under UNIX, you can use the following command to display the maximum number of open files in the operating system shell.

```
ulimit -n
```

This value should be greater than the expected number of TCP/IP connections.

4 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, publishers and subscribers 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/OSD	File *ETB-ATTR* in library *EXX951.JOBS*.
z/VSE	Library member *ETBnnn.ATR*, where ETB*nnn* is 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.
- Multiple topics can be included in a single topic definition section. The attribute settings will apply to all topics defined in the section.

- Attributes specified after the service definition (`CLASS`, `SERVER`, `SERVICE` *keywords*) overwrite the default characteristics for the service.
- Attributes specified after the topic definition (`TOPIC` *keyword*) override the default characteristics for the topic.
- 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.

Attribute	Values	Opt/ Req	Operating System				
			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		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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	See also *Accounting in EntireX Broker* in the z/OS administration documentation.						
ACCOUNTING-VERSION	1 | 2 | 3 | 4	O	z	u	w		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. This parameter applies to z/OS, UNIX, Windows and BS2000/OSD when ACCOUNTING is activated.						
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.						
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.						

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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*. 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. If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ NUM-PUBLISHER ■ LONG-BUFFER-DEFAULT ■ NUM-SERVER ■ PUBLICATION-DEFAULT ■ NUM-SERVICE-EXTENSION ■ SERVER-DEFAULT ■ NUM-SERVICE ■ SHORT-BUFFER-DEFAULT ■ NUM-SHORT[-BUFFER] ■ SUBSCRIBER-DEFAULT ■ NUM-SUBSCRIBER-TOTAL ■ NUM-CLIENT ■ NUM-SUBSCRIBER ■ NUM-CMDLOG-FILTER ■ NUM-TOPIC-EXTENSION ■ NUM-COMBUF ■ NUM-TOPIC-TOTAL						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	■ NUM-CONV[ERSATION] ■ NUM-TOPIC ■ NUM-LONG[-BUFFER] ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-PUBLICATION ■ 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*.						
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:**						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	1. If broker is started twice, the second instance will kill the first by removing the IPC resources. 2. For BS2000/OSD, z/OS and z/VSE, see separate attribute `FORCE` in section *Adabas SVC/Entire Net-Work-specific Attributes*.						
`HEAP-SIZE`	1024 *n*	O	z	u	w	v	b
	Defines the size of the internal heap in KB. We strongly recommend using the default value (1024 KB).						
`ICU-CONVERSION`	YES | NO	O	z	u	w	v	b
	Disable or enable ICU conversion. YES ICU is loaded and available for conversion. It is a prerequisite for SAGTCHA and SAGTRPC. NO ICU is not loaded and not available for conversion. SAGTCHA and SAGTRPC cannot be used. If any of the broker service definitions uses the internationalization approach “ICU conversion”, that is, the conversion methods SAGTCHA and SAGTRPC are defined by the service-specific or topic-specific attribute `CONVERSION`, `ICU-CONVERSION` must be set to "YES". The internationalization approaches “Translation”, “Translation User Exit” and “SAGTRPC User Exit” do not require ICU conversion. If all broker service definitions use these internationalization approaches, `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-SET-DATA-DIRECTORY`	YES | NO	O		u	w		
	Disable or enable ICU custom converter usage. Not defined for mainframe platforms. 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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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 or topic. 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 (or publication).						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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 under *Broker CIS Data Structures* in the ACI Programming documentation. 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
	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	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	attribute CMDLOG is set to "YES". See *Command Logging in EntireX* for more information.						
NUM-COMBUF	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 Definition***.						
NUM-LONG-BUFFER or NUM-LONG	*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 Definition***.						
NUM-PUBLICATION	*n* | AUTO	O	z	u	w	v	b
	Defines the number of publications that can be active concurrently. *n* Number of publications AUTO Uses the PUBLICATION-DEFAULT and the topic-specific PUBLICATION-LIMIT to calculate the number of publications. The values used in the calculation must not be set to "UNLIM" **Note:** 1. A value of 0 (zero) is invalid. 2. If a wildcard topic is defined in the topic-specific section of the attribute file, the value of AUTO is invalid.						
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-PUBLISHER	*n*	O	z	u	w	v	b
	Number of publishers that can access the broker concurrently. A value of 0 (zero) is invalid.						
NUM-SERVER	*n* | AUTO	R	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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 Definition*.							
NUM-SERVICE	*n*	R	z	u	w	v	b
	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							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	considers the value of SERVER-DEFAULT and even the individual SERVER-LIMIT for each service definition (see note below).						
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. 2. In *conversational* 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 ***Wildcard Service Definition***.						
NUM-SUBSCRIBER	*n* | AUTO	O	z	u	w	v	b
	Defines the number of subscribers that can be active concurrently. *n* Number of subscribers. AUTO Uses the SUBSCRIBER-DEFAULT and the topic-specific SUBSCRIBER-LIMIT to calculate the number of subscribers. A value of 0 (zero) is invalid. If a wildcard topic is defined in the topic-specific section of the attribute file, the value of AUTO is invalid.						
NUM-SUBSCRIBER-TOTAL	*n* | AUTO	O	z	u	w	v	b
	Defines the total number of subscribers that can be durably subscribed. Their subscription information is saved in the persistent store. *n* Total number of subscribers.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	AUTO Uses the value defined or calculated for NUM-SUBSCRIBER. A value of 0 (zero) is invalid. This value must be greater than or equal to the NUM-SUBSCRIBER value. Parameter is required if SUBSCRIBER-STORE=PSTORE is defined.						
NUM-TOPIC	*n*	O	z	u	w	v	b
	Defines the number of topics that can be active in the broker. A value of 0 (zero) is invalid.						
NUM-TOPIC-EXTENSION	*n* | AUTO	O	z	u	w	v	b
	Defines the number of topic extensions to link subscribers to topics. *n* Number of topic extensions. AUTO Uses the value specified for NUM-SUBSCRIBER + NUM-PUBLISHER, plus an extra cushion. *not specified* If this attribute is not set, the default value is NUM-SUBSCRIBER multiplied by NUM-TOPIC. The minimum value is NUM-SUBSCRIBER. The maximum value is NUM-SUBSCRIBER multiplied by NUM-TOPIC. Caution is recommended with this attribute. ■ Set this attribute only if the storage resources allocated for topic extensions need to be restricted. ■ Note that the value <*n*> allows only the specified number of topic instances of <*n*> to be used. ■ Value AUTO calculates the number of allowed server instances from NUM-SUBSCRIBER, which itself might set to AUTO. In this case, this also considers the value of SERVER-DEFAULT and even the individual SERVER-LIMIT for each topic definition (see note below).						
NUM-TOPIC-TOTAL	*n* | AUTO	O	z	u	w	v	b
	Defines the total number of topics for which durable subscribers are allowed. *n* Total number of topics that allow durable subscriptions. AUTO Uses the value defined for NUM-TOPIC. This value must be greater than or equal to the NUM-TOPIC value. This parameter is required if SUBSCRIBER-STORE=PSTORE is defined.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
NUM-UOW	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 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.						
NUM-WORKER	1 | *n* (max. 10)	R	z	u	w	v	b
	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.						
NUM-WQE	1 - 32768	R	z	u	w	v	b
	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.						
PARTICIPANT-BLACKLIST	YES | NO	R	z	u	w	v	b
	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 *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific broker administration documentation.						
PARTNER-CLUSTER-ADDRESS	A32	R	z	u	w	v	b
	This is the address of the load/unload broker in transport-method-style. Transport methods TCP and SSL are supported. See *Transport-method-style Broker ID* for more details. This attribute is required if the attribute RUN-MODE is specified.						
POLL	YES | NO	O	z	u			
	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 and UNIX.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	YES The `poll()` system call is used to lift the resource restrictions with `select()` in multiplexing file descriptor sets. 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.						
`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. NO Do not create the PSTORE report file. YES Create the PSTORE report file. See also ***Persistent Store Report***.						
`PSTORE-TYPE`	DIV (z/OS) | CTREE (UNIX, Windows) | Adabas (all platforms) | FILE (UNIX, Windows)	O	z	u	w	v	b
	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* in the z/OS administration documentation.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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.						
PSTORE-VERSION	2 | 3 | 4	O	z	u	w	v	b
	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. We recommended setting PSTORE-VERSION=3. PSTORE-VERSION=4 is needed to use the DIV PSTORE handler introduced with version 9.0. It requires much less configuration data. **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.						
PUBLICATION-DEFAULT	n | UNLIM	O	z	u	w	v	b
	Default number of publications that are allocated for every topic. *n* Number of publications. UNLIM The number of publications is restricted only by the number of publications globally available. Precludes the use of NUM-PUBLICATION=AUTO. This value can be overridden by specifying a PUBLICATION-LIMIT for the topic. A value of 0 (zero) is invalid.						
PUBLICATION-LIFETIME	*n* | *n*S | *n*M | *n*H | *n*D | *n*Y	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Lifetime of a publication in absolute time units. Publications are retained by broker until they are either received by all subscribers or the publication lifetime has expired. *n* Same as *n*S. *n*S Publication lifetime in seconds (max. 2147483647). *n*M Publication lifetime in minutes (max. 35791394). *n*H Publication lifetime in hours (max. 596523). *n*D Publication lifetime in days (max. 24855). *n*Y Publication lifetime in years (max. 68). The publication lifetime is calculated even for periods of time when broker is stopped.						
PUBLISH-AND-SUBSCRIBE	YES | NO	O	z	u	w	v	b
	Run publish and subscribe subsystem. Subsystem requires a license.						
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 the EntireX Broker security exits are activated. NO The security exits are not activated. YES The security exits are activated. If the security routines cannot be activated, the broker will not start. Broker trace reports the type of security which is active and from where the security module USRSEC is loaded:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	■ EntireX Security ■ User-written USRSEC.						
SECURITY-PATH	A255	O	z	u	w		b

	Full path and file name of an executable file (for example, DLL for Windows or shared library for UNIX) containing the user security exit which the kernel will load and call. Example:
	`SECURITY-PATH=usersec.dll`
	This assumes the DLL is in the default path. Or:
	`SECURITY-PATH=c:\brokerexit\yoursecu.dll`
	If the path name contains spaces, enclose it in quotation marks. Example:
	`SECURITY-PATH="c:\Software AG\broker exit\yoursecu.dll"`
	Note: This attribute is used only when implementing a user-written security exit.						

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Number of short buffers to be allocated for each 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. *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.						
SSLPORT	See PORT.						
SSL-RESTART	See RESTART.						
SSL-RETRY-LIMIT	See RETRY-LIMIT.						
SSL-RETRY-TIME	See RETRY-TIME.						
SSTORE SSTORE-TYPE	These parameters are obsolete. The subscriber store in a secondary store is no longer supported. We recommend you use the PSTORE persistent store to store your subscriber data. For this, set broker-specific parameter SUBSCRIBER-STORE=PSTORE.						
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* under ***Broker Resource Allocation***.						
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.						
SUBSCRIBER-DEFAULT	*n* | UNLIM	O	z	u	w	v	b
	Default number of subscribers that are allowed for every topic. *n* Number of subscribers UNLIM The number of subscribers is restricted only by the number of subscribers globally available. Precludes the use of NUM-SUBSCRIBER=AUTO.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This value can be overridden by specifying a SUBSCRIBER-LIMIT for the topic. A value of 0 (zero) is invalid.						
SUBSCRIBER-STORE	NO | PSTORE	O	z	u	w	v	b
	Determines whether subscriber information is stored and where. NO No subscriber information is to be stored. PSTORE Save subscriber data in PSTORE. **Tip:** The subscriber store in a secondary store is no longer supported. We recommend you use the PSTORE persistent store to store your subscriber data.						
TCPPORT	See PORT.						
SWAP-OUT-NEW-UOWS	NO | YES	O	z	u	w	v	b
	Determines whether conversations with units of work remain in memory or are swapped. See slso *Swapping out New Units of Work*. NO All conversations with UOWs remain in memory. YES Conversations with UOWs (STORE=BROKER) created by a client and finished with an EOC without being accepted by a server will be swapped out of memory. The data is persisted on PSTORE and there is no need to keep it in memory unless a server wants to receive this data. **Note:** See service-specific attribute MIN-UOW-CONVERSATIONS-IN-MEMORY for defining a minimum number of UOW conversations kept in memory to improve the performance for servers receiving new UOW conversations without waiting for swap-in of data from PSTORE. During broker restart, all new and unassigned UOW conversations remain in PSTORE only. This reduces the restart time significantly. See also *Swapping out New Units of Work*.						
TCP-RESTART	See RESTART.						
TCP-RETRY-LIMIT	See RETRY-LIMIT.						
TCP-RETRY-TIME	See RETRY-TIME.						
TOPIC-UPDATES	YES | NO	O	z	u	w	v	b
	Switch on/off automatic update of topic defaults in the broker. YES The broker reads the attribute file whenever a topic is being subscribed for the first time. This allows broker to honor modifications in the						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	attribute file without a restart. The attribute file is read only when the first subscriber subscribes to a particular topic. It is not reread when a second subscriber subscribes to the same topic. 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.						
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:
	
	`TRACE-DD = "DSNAME=EXX.GDG,`
	`DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB),`
	`DISP=(NEW,CATLG,CATLG),`
	`SPACE=(CYL,(100,10)),`
	`STORCLAS=SMS"`						

TRACE-LEVEL	0 - 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 if SAGTRPC is used for CONVERSION with the conversion options SUBSTITUTE-NONCONV or STOP.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. 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 restarting the broker, use System Management Hub or ETBCMD. Trace levels 2, 3, and 4 should be used only when requested by Software AG support.						
TRANSPORT	TCP | SSL | NET	O	z	u	w	v	b
	The broker transport may be specified as any combination of one or more of the following methods: TCP TCP/IP is supported. SSL SSL or 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. 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 (or TLS), and Entire Net-Work transport methods will be supported by the broker. Section ***TCP/IP-specific Attributes*** describes the parameters for each transport method.						
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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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-MSGS	See MAX-MESSAGES-IN-UOW.						
UWSTAT-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	v	b
	The value to be added to the UWTIME (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). *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 UWSTAT-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 UWTIME.						
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 UWTIME 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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	1 - 254 Multiplied by the value of UWTIME to determine how long a persistent status will be retained. **Note:** This attribute has not been supported since EntireX version 7.3. Use UWSTAT-LIFETIME instead.						
UWTIME	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*.						
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. NO If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, 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 communications with the PSTORE. Broker will reject any user requests until broker is able to contact the Adabas database.						
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 | nH	O	z	u	w		b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*S.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	*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 Definition* and *Service Update Modes* below the table.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
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. 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>).						
CONV-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	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						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	the use of NUM-CONVERSATION=AUTO in the Broker section of the attribute file. *n* 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.							
CONV-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Non-activity time for connections. *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 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.							
CONVERSION	Format: A255 (SAGTCHA [, TRACE =*n*] [, *OPTION* =*s*] | SAGTRPC [, TRACE =*n*] [, *OPTION* =*s*] | *name* [, TRACE =*n*] | NO)	O	z	u	w	v	b
	Defines conversion for internationalization. See *Internationalization with EntireX* and *What is the Best Internationalization Approach to use?* under *Introduction to Internationalization* for help on making decisions about the internationalization approach.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	SAGTCHA — Conversion using ICU Conversion [1] for *ACI-based Programming*. SAGTRPC [2] — Conversion using ICU Conversion [1] for *RPC-based Components* and *Reliable RPC*. We recommend always using SAGTRPC for RPC data streams. *Conversion with Multibyte, Double-Byte and other Complex Codepages* will always be correct, and *Conversion with Single-byte Codepages* is also efficient because SAGTRPC detects single-byte codepages automatically. See *Conversion Details*. *<name>* [2] — Name of the SAGTRPC user exit for RPC-based components. See also *Configuring SAGTRPC User Exits* under *Configuring Broker for Internationalization* in the platform-specific administration documentation and *Writing SAGTRPC User Exits* 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. Only one internationalization approach can be active at one time for a service. The `CONVERSION` attribute for internationalization 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 and SAGTRPC user exit are not supported on z/VSE. **TRACE** If tracing is switched on, the trace output is written to the broker log file: 0 No tracing						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	1 Trace level 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 Trace level ADVANCED — Tracing of incoming, outgoing parameters and the payload. 3 Trace level 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	NO | 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.						
ENCRYPTION-LEVEL	0 | 1 | 2	O	z	u	w	v	b
	Enforce encryption when data is transferred between client and server. 0 No encryption is enforced. 1 Encryption is enforced between server and broker kernel. 2 Encryption is enforced between server and broker kernel, and also between client and broker. See also ENCRYPTION-LEVEL in Broker ACI control block and *Encryption* under *Writing Applications using EntireX Security* in the ACI Programming documentation. **Note:** The per service ENCRYPTION-LEVEL attribute is to be specified only where the broker attribute SECURITY=YES has been specified and only if you are using EntireX Security.						
LOAD-BALANCING	YES | 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	v	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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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.							
MIN-UOW-CONVERSATIONS-IN-MEMORY	256 | *n*	O	z	u	w	v	b
	Defines the minimum number of UOW conversations (STORE=BROKER, created by a client and finished with an EOC without being accepted by a server) kept in memory to improve the performance for servers receiving new UOW conversations without waiting for data to be swapped in from PSTORE. See also *Swapping out New Units of Work*. 256 The default value should be used if producer (client) and consumer (server) of UOW conversations are both active at the same time regardless of the speed producing or consuming UOW conversations. It guarantees a reasonable balance between memory being used and swap-out/swap-in activities. *n* Minimum number of UOW conversations kept in memory. The value *n* is equal to or greater than 256. **Note:** If broker-specific attribute SWAP-OUT-NEW-UOWS is set to "NO", MIN-UOW-CONVERSATIONS-IN-MEMORY has no effect.							
MUOW	See MAX-UOWS.							
NOTIFY-EOC	NO | YES	O	z	u	w	v	b
	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.							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. **Caution:** 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`.						
`NUM-UOW`	Alias for `MAX-UOWS`.						
`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.						

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. This attribute can be overridden by the STORE field in the Broker ACI control block.						
TRANSLATION	Format: A255 SAGTCHA | NO | <*name*>	O	z	u	w	v	b
	Activates *translation* or *translation user exit* for internationalization (see *Translation User Exit* under *Introduction to Internationalization*). For help on deciding the right internationalization approach for your environment, see *What is the Best Internationalization Approach to use?* under *Introduction to Internationalization* SAGTCHA Conversion routine SAGTCHA for *ACI-based Programming*, *RPC-based Components* and *Reliable RPC*. 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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	The CONVERSION attribute for internationalization 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-MSGS	Alias for MAX-MESSAGES-IN-UOW.						
UWSTAT-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	v	b
	The value to be added to the UWTIME (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). *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 UWSTAT-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 UWTIME.						
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 UWTIME 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 UWTIME to determine how long a persistent status will be retained.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Note: This attribute has not been supported since EntireX version 7.3. Use `UWSTAT-LIFETIME` instead.						
`UWTIME`	1D | *n* S | *n* M | *n* H | nD	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 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.						

Wildcard Service Definition

The special names of `CLASS = *`, `SERVER = *` and `SERVICE = *` are allowed in the service-specific section 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 can not 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).

The situations 1 and 2 above are reported to the broker log file if 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	yes	yes	No message.	No message

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)
	codepage-dependent default replacement character.				
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.

Topic-specific Attributes

The topic-specific attribute section begins with the keyword `DEFAULTS=TOPIC` as shown in the sample attribute file. It contains attributes that apply to the publish and subscribe communication model.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
`ALLOW-DURABLE`	YES | NO	O	z	u	w	v	b
	Determines whether a subscriber is allowed to perform a durable subscription to a topic. YES Subscriber may perform durable subscription. NO Durable subscription not allowed. If users are allowed to durably subscribe to any topic, you must specify a value for the `SUBSCRIBER-STORE` parameter.						
`ALLOW-USER-SUBSCRIBE`	YES | NO	O	z	u	w	v	b
	Determines if it is possible for a user to subscribe to a topic directly (YES) or only by Administrator. YES Users are allowed to subscribe to the topic. NO Users must be subscribed by the Administrator through CIS. See *Broker Command and Information Services*. The subscribe request of users is rejected.						
`AUTO-COMMIT-FOR-SUBSCRIBER`	NO | YES	O	z	u	w	v	b
	NO No `COMMIT` performed. YES An implicit `COMMIT` is performed by broker when the subscriber receives a publication, that is, the subscriber does not need the `CONTROL_PUBLICATION` option `COMMIT` after receiving each publication. **Caution:** You may lose your last message.						
`CONVERSION`	Format: A255 `( SAGTCHA [TRACE =`*n*`]`	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	[, *OPTION* =*s*])						

Defines conversion for internationalization. See *Internationalization with EntireX*. For help on making decisions about the internationalization approach, see *What is the Best Internationalization Approach to use?* under *Introduction to Internationalization*

SAGTCHA	Conversion using ICU Conversion for *ACI-based Programming*. For more information see *Conversion Details*. See also *Configuring ICU Conversion* 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.

Only one internationalization approach can be active at one time for a topic. The `CONVERSION` attribute for internationalization overrides the `TRANSLATION` attribute when defined for a topic, that is, when `TRANSLATION` and `CONVERSION` are both defined, `TRANSLATION` will be ignored.

TRACE

If tracing is switched on, the trace output is written to the broker log file:

0 No tracing	
1 Trace level `STANDARD`	This level is an "on-error" trace. It provides information on conversion errors only. Please note that if ***`OPTION` Values for Conversion*** are set, errors are ignored.
2 Trace level `ADVANCED`	Tracing of incoming, outgoing parameters and the payload.
3 Trace level `SUPPORT`	This trace level is for support diagnostics and should only be switched on when requested by Software AG support.

OPTION

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	See *OPTION Values for Conversion* under *Service-specific Attributes* above.						
LONG-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of long message buffers for the topic. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Excludes 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 topic section. A value must be specified or the LONG-BUFFER-LIMIT attribute must be suppressed entirely for the topic so that the default (LONG-BUFFER-DEFAULT) becomes active.						
MAX-MESSAGES-IN-PUBLICATION	16 | *n*	O	z	u	w	v	b
	Maximum number of messages in a publication.						
MAX-PUBLICATION-MESSAGE-LENGTH	31647 | *n*	O	z	u	w	v	b
	Maximum size of a message in a publication. The actual publication size is transport-dependent.						
PUBLICATION-LIFETIME	*n* | *n*S | *n*M | *n*H | *n*D | *n*Y	O	z	u	w	v	b
	Lifetime of a publication in absolute time units. Publications are retained by broker until they are either received by all subscribers or the publication lifetime has expired. *n* Same as *n*S. *n*S Publication lifetime in seconds (max. 2147483647). *n*M Publication lifetime in minutes (max. 35791394). *n*H Publication lifetime in hours (max. 596523). *n*D Publication lifetime in days (max. 24855). *n*Y Publication lifetime in years (max. 68). The publication lifetime is calculated even for periods of time when broker is stopped.						
PUBLICATION-LIMIT	*n* | UNLIM	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	There is no default. Maximum number of publications possible for this topic. If specified, this overrides the publication default value, which is a general maximum value per topic. If neither parameter is specified, the total number of publications for the topic is limited only by NUM-PUBLICATION. *n* Number of publications. UNLIM The number of publications is restricted only by the number of publications globally available. Excludes the use of NUM-PUBLICATION=AUTO in the Broker section of the attribute file. A value of 0 (zero) is invalid. If PUBLICATION-LIMIT=AUTO is specified in the Broker section of the attribute file, PUBLICATION-LIMIT=UNLIM is not allowed in the topic section. A value must be specified, or the PUBLICATION-LIMIT attribute must be suppressed entirely for the topic so that the default (PUBLICATION-DEFAULT) becomes active.						
PUBLISHER-NONACT	5M | *n* | *n*S | *n*M | *n*H | *n*D | *n*Y	O	z	u	w	v	b
	Non-activity of the publisher, after which an auto-logoff is performed and the publisher's resources 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). *n*D Non-activity time in days (max. 24855). *n*Y Non-activity time in years (max. 68). If not specified, defaults to 5 minutes. This is the time after which the publisher's internal memory structures will be cleaned up and a subsequent logon is required.						
SHORT-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of short message buffers for the topic. UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Excludes the						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	use of NUM-LONG-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of short message buffers. A value of 0 (zero) is invalid. If NUM-SHORT-BUFFER=AUTO is specified in the Broker section of the attribute file, SHORT-BUFFER-LIMIT=UNLIM is not allowed in the topics section. A value must be specified, or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the topic so that the default (SHORT-BUFFER-DEFAULT) becomes active.							
SSTORE SSTORE-TYPE	These parameters are obsolete. The subscriber store in a secondary store is no longer supported. We recommend you use the primary persistent store (PSTORE) to store your subscriber data. For this, set broker-specific parameter SUBSCRIBER-STORE=PSTORE.						
SUBSCRIBER-LIMIT	*n* | UNLIM	O	z	u	w	v	b
	There is no default. Maximum number of subscriptions possible for this topic. If specified, this overrides the subscriber default value, which is a general maximum value per topic. If neither parameter is specified, the total number of subscribers for the topic is limited only by NUM-SUBSCRIBER. *n* Number of subscribers. UNLIM The number of subscribers is restricted only by the number of subscribers globally available. Excludes the use of NUM-SUBSCRIBER=AUTO in the Broker section of the attribute file. A value of 0 (zero) is invalid. If NUM-SUBSCRIBER=AUTO is specified in the Broker section of the attribute file, SUBSCRIBER-LIMIT=UNLIM is not allowed in the topic section. A value must be specified, or the SUBSCRIBER-LIMIT attribute must be suppressed entirely for the topic so that the default (SUBSCRIBER-DEFAULT) becomes active.							
SUBSCRIBER-NONACT	5M | *n* | *n*S | *n*M | *n*H | *n*D | *n*Y	O	z	u	w	v	b
	Non-activity of the subscriber after which an auto-logoff is performed and the publisher's resources are freed. *n* Same as *n*S.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	*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). *n*D Non-activity time in days (max. 24855). *n*Y Non-activity time in years (max. 68). In the case of a non-durable subscriber, the user's subscription is also cancelled. In the case of a durable subscriber, the user's subscription is persisted, and it is not necessary for the user to issue any subsequent `SUBSCRIBE` commands. The subscription of a durable subscriber is also persisted even while broker is stopped. If not specified, defaults to 5 minutes. This is the time after which the subscriber's internal memory structures will be cleaned up and a subsequent logon is required.							
`SUBSCRIPTION-EXPIRATION`	NEVER | *n* | *n*S | *n*M | *n*H | *n*D | *n*Y	O	z	u	w	v	b
	Lifetime of a user's subscription in absolute time units. Subscriptions are retained by broker until either the user issues an `UNSUBSCRIBE` command or the subscription lifetime has expired. NEVER Subscriber will never be purged from PSTORE. *n* Same as *n*S. *n*S Expiration time in seconds (max. 2147483647). *n*M Expiration time in minutes (max. 35791394). *n*H Expiration time in hours (max. 596523). *n*D Expiration time in days (max. 24855). *n*Y Expiration time in years (max. 68). Durable subscriptions remain effective even if the user performs the `LOGOFF` command or broker is stopped. The subscription lifetime is calculated also for periods of time when broker is stopped. `SUBSCRIPTION-EXPIRATION` is the time after which the subscription expires. In the case of durable subscription, the subscription is removed from the PSTORE. Broker removes expired subscriptions only when the user is not currently active, for example							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	when the user has issued a LOGOFF command or after the SUBSCRIBER-NONACT has passed if no LOGOFF is issued. If SUBSCRIBER-NONACT is specified greater than SUBSCRIPTION-EXPIRATION, broker adjusts SUBSCRIPTION-EXPIRATION to the value of SUBSCRIBER-NONACT.						
TOPIC	A96 (case-sensitive)	R	z	u	w	v	b
	Name of the topic for publish and subscribe processing. Valid characters for topic name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma.						
TRANSLATION	Format: A255 SAGTCHA | NO | <*name*>	O	z	u	w	v	b
	Activates *translation* or *translation user exit* for internationalization (see *Translation User Exit* under *Introduction to Internationalization*). See also *What is the Best Internationalization Approach to use?* under *Introduction to Internationalization* SAGTCHA — Conversion routine SAGTCHA for ACI-based programming, RPC-based components and for *Reliable RPC*. 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 SAGTRPC User Exits* under *Configuring Broker for Internationalization* in the platform-specific administration documentation and *Writing SAGTRPC User Exits* in the platform-specific administration documentation. The CONVERSION attribute for internationalization overrides the TRANSLATION attribute when defined for a service, i.e. when TRANSLATION and CONVERSION are both defined, TRANSLATION will be ignored.						

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 the internationalization approaches ICU conversion and SAGTRPC user exit. These attributes do not apply to other approaches. See *Internationalization with EntireX* for more information.

Attribute	Values	Opt/ Req	Operating System				
			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, publisher or subscriber). See *Broker's Locale String Defaults* under *Locale String Mapping* in the internationalization documentation. 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.), and
	■ one of the internationalization approaches ICU conversion or SAGTRPC user exit is used. See *ICU Conversion* under *Introduction to Internationalization* and *SAGTRPC User Exit* under *Introduction to Internationalization*.
	Example:
	`DEFAULTS=CODEPAGE`
	`  /* Broker Locale String Defaults */`
	`  DEFAULT_ASCII=windows-950`
	For more examples, see *Configuring Broker's Locale String Defaults* under *Locale String Mapping* in the internationalization documentation and also ***Additional Notes*** below.						

`DEFAULT_EBCDIC_IBM`	Any ICU converter	O	z	u	w	v	b

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

	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server, publisher or subscriber). See *Broker's Locale String Defaults* under *Locale String Mapping* in the internationalization documentation. 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.) and
	■ one of the internationalization approaches ICU conversion or SAGTRPC user exit is used.
	Example:
	`DEFAULT=CODEPAGE`
	`  DEFAULT_EBCDIC_IBM=ibm-937`
	For more examples, see *Configuring Broker's Locale String Defaults* under *Locale String Mapping* 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, publisher or subscriber). See *Broker's Locale String Defaults* under *Locale String Mapping* in the internationalization documentation. 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/OSD), and ■ one of the internationalization approaches ICU conversion or SAGTRPC user exit is used. Example:						

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

	`DEFAULT=CODEPAGE`
	`DEFAULT_EBCDIC_SNI= bs2000-edf03drv`
	
	For more examples, see *Configuring Broker's Locale String Defaults* under *Locale String Mapping* 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* under *Locale String Mapping* in the internationalization documentation. 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, publisher or subscriber) 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* under *Locale String Mapping* in the internationalization documentation. Example:
	
	`DEFAULTS=CODEPAGE`
	`/* Broker Locale String Codepage Assignments */`
	`ASCII=ISO8859`
	`EUC_JP_LINUX=ibm-33722_P12A-1999`
	`/* Customer-written ICU converters */`
	`CP1140=myebcdic`
	`CP0819=myascii`						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	For more examples, see *Bypassing Broker's Built-in Locale String Mapping* under *Locale String Mapping* in the internationalization documentation 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 the internationalization approach and if the style in not known by ICU, e.g. ECS*nnnn*, <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* under *Locale String Mapping* in the internationalization documentation. For more details on ICU and ICU converter name standards, see *ICU Resources* under *Introduction to Internationalization*.
- If SAGTRPC user exit is used for the internationalization approach, 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* under *Locale String Mapping* in the internationalization documentation.
- See `CONVERSION` and `CONVERSION` attribute `CONVERSION` on this page for the internationalization approach 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.

Attribute	Values	Opt/ Req	Operating System				
			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/OSD.						
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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
IDTNAME	FORMAT: A8 *idtname* | 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. The ID table is used to perform various internal functions, including communication between the caller program and the EntireX Broker. Only supported under BS2000/OSD.						
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
	Specifies whether the broker ID is local. NO Broker ID can be accessed from remote nodes. YES The broker ID is local. It is not accessible from remote nodes.						
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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. 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 any of the following application components using this transport mechanism: ■ clients ■ servers ■ publishers ■ subscribers						
NODE	1-65534	O	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. Please note that the maximum value for NODE that is allowed for Entire Net-Work under UNIX is 255. If NODE is specified, it overrides the DBID derived from the numeric part of BROKER-ID.						
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				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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. 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 restarting the broker, use System Management Hub or ETBCMD. Trace levels 2, 3, and 4 should be used only when requested by Software AG support.						

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/OSD. This does not require broker to be running under TSOS. See *EntireX Broker Security Server for BS2000/OSD* in the BS2000/OSD administration documentation.							
`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.							

`AUTHENTICATION-TYPE`	OS | *`ldapUrl`* | *`iafUrl`*	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. *`ldapUrl`* Authentication is performed against the LDAP repository specified under *`ldapUrl`*. Not supported under BS2000/OSD. ■ For TCP, specify repository URL:							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	AUTHENTICATION-TYPE="ldap://*HostName* [:*PortNumber*]" ■ For SSL or TLS: AUTHENTICATION-TYPE="ldaps://*HostName* [:*PortNumber*]" If no port number is specified, the default is the standard LDAP port numbe 389 for TCP transport. Examples for TCP and SSL (or TLS): AUTHENTICATION-TYPE="ldap://myhost.mydomain.com" AUTHENTICATION-TYPE="ldaps://myhost.mydomain.com:636"						
	iafUrl Authentication is performed using Software AG's Integrated Authentication Framework against the IAF service specified under *iafUrl*. Not supported ur BS2000/OSD. The URL of the IAF service is specified using AUTHENTICATION-TYPE= "iaf://HostName[:PortNumber]?SSLParameters" If no port number is specified, the default is port number 1958. SSL or TLS parameters are specified in the same format as for the ACI function SETSSLPAR Example: AUTHENTICATION-TYPE="iaf://myhost.mydomain.com:1000 AUTHENTICATION-TYPE= "iaf://myhost.mydomain.com:10000? verify_server= no& trust_store= /opt/softwareag/EntireX/etc/ExxCACert.pem" On z/OS, the URL of an IAF service running on the same host may specified						

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

	`AUTHENTICATION-TYPE= "iaf.ipc://IAFServiceID[:SVCNumber]"`
	
	Example:
	
	`AUTHENTICATION-TYPE= "iaf.ipc://IAF075:SVC245"`						

`AUTHORIZATIONDEFAULT`	YES | NO	O		u	w		
	Determines whether access is granted to a specified service if the specified could not be found listed in the repository of authorization rules. 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 and `AUTHORIZATIONDEFAULT` to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Administering Authorization Rules using System Management Hub* in the UNIX and Windows administration documentation.						
`AUTHORIZATIONRULE`	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 `AUTHORIZATIONRULE` entries in the attribute file is 16. Applies only when using EntireX Security under UNIX or Windows. Authorization rules can be stored within a repository. When an authorization call occurs, EntireX Security uses the values of this parameter and `AUTHORIZATIONDEFAULT` to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Administering Authorization Rules using System Management Hub* in the UNIX and Windows administration documentation.						
`CHECK-IP-ADDRESS`	YES | NO	O	z				
	Determines whether the TCP/IP address of the caller is subject to a resource check.						
`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*						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	(Optional) under *Installing EntireX Security under z/OS under z/OS* in the z/OS installatio documentation.						
FACILITY-CHECK	NO | YES	O	z				
	It is possible to check whether a particular user is at all allowed to use an application be performing a password check. The advantage of this additional check is that when the is not allowed to use this application, the broker returns error 00080013 and does not t authenticate the user. Failing an authentication check may lead to the user's password b 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 be 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 ca						
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-PERSON-BASE-BINDDN	*ldapDn*	O	z	u	w		
	Used with LDAP authentication to specify the distinguished name where authenticati information is stored. This value is prefixed with the user ID field name (see below). Exam LDAP-PERSON-BASE-BINDDN="cn=users,dc=mydomain,dc=com"						
LDAP-REPOSITORY-TYPE	OpenLDAP | ActiveDirectory | SunOneDirectory | Tivoli | Novell | ApacheDS	O	z	u	w		
	Use predefined known fields for the respective repository type. Specify the repository that most closely matches your actual repository. In the case of Windows Active Direc 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 authentication check. In practice, this determines whether or not the password supplie the user is passed in plain text between the broker kernel and the LDAP server. If SAS activated, this implies that the password is encrypted.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	NO Password is sent to LDAP server in plain text. YES Password is sent to LDAP server encrypted.						
LDAP-USERID-FIELD	cn | *uidFieldName*	O	z	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. This parameter defaults to 80 if a value is not specified.						
PASSWORD-TO-UPPER-CASE	NO | YES	O	z	u	w		b
	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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SECURITY-LEVEL	AUTHORIZATION | AUTHENTICATION | ENCRYPTION	O	z	u	w	v	b
	Specifies the mode of operation. AUTHORIZATION — Authorization, authentication, and encryption (not under BS2000/OSD or z/VSE). AUTHENTICATION — Authentication and encryption. ENCRYPTION — Encryption only. **Caution:** 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 importa distinguish between production, test, and development environments. YES — This causes the broker ID to be used as a prefix for all authorization checks. *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 defau behavior).						
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 attribu file.						
TRUSTED-USERID	YES | NO	O	z				
	Activates the trusted user ID mechanism for broker requests arriving over the local Ad IPC mechanism.						
USERID-TO-UPPER-CASE	NO | YES	O	z				b
	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, 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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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
	The TCP/IP port number on which the broker will listen for connection requests. If specified, PORT overrides broker attribute TCPPORT. **Note:** TCPPORT will be retired with the next version. If PORT is not specified but TCPPORT is specified, TCPPORT is used. If TCPPORT is not specified, the broker will attempt to find its TCP/IP port number from the TCP/IP Services file, using *getservbyname*. If broker cannot find its TCP/IP port number from the TCP/IP Services file, it will use the default value of 1971. A maximum of five HOST/PORT pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.						
RESTART	YES | NO	O	z	u	w	v	b
	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. If specified, RESTART overrides broker attribute TCP-RESTART. **Note:** TCP-RESTART will be retired with the next version. If RESTART is not specified but TCP-RESTART is specified, TCP-RESTART is used. The RESTART setting applies to all TCP/IP communicators.						
RETRY-LIMIT	20 | *n* | UNLIM	O	z	u	w	v	b
	Maximum number of attempts to restart the TCP/IP communicator. If specified, RETRY-LIMIT overrides broker attribute TCP-RETRY-LIMIT. **Note:** TCP-RETRY-LIMIT will be retired with the next version. If RETRY-LIMIT is not specified but TCP-RETRY-LIMIT is specified, TCP-RETRY-LIMIT is used. The RETRY-LIMIT setting applies to all TCP/IP communicators.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
RETRY-TIME	3M | *n* | *n*S | *n*M | *n*H	O	z	u	w	v	b
	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. If specified, RETRY-TIME overrides broker attribute TCP-RETRY-TIME. **Note:** TCP-RETRY-TIME will be retired with the next version. If RETRY-TIME is not specified but TCP-RETRY-TIME is specified, TCP-RETRY-TIME is used. The RETRY-TIME 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		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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. 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 restarting the broker, use System Management Hub or `ETBCMD`. Trace levels 2, 3, and 4 should be used only when requested by Software AG support.						

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/OSD, z/VSE.

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
TRACE-LEVEL	0-8	O		u	w		
	Trace level for c-tree persistent store. It overrides the global value of trace level in the attribute file.						

SSL-specific Attributes

The SSL-specific attribute section 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. In this section, "SSL" also applies to TLS (Transport Layer Security).

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 implementation. SSL 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 stack; others are optional. When an SSL 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 server side (the broker always implements the server side). Ths stubs connect to the broker and thereby become the SSL clients.

Under UNIX and Windows, the OpenSSL implentation of the SSL server side is used; on z/OS and BS2000/OSD it is GSK.

Example for OpenSSL:

`CIPHER-SUITE=RC4-MD5`	Use RC4 with standard 128-bit key and MD5 as hash.
`CIPHER-SUITE=EXP-EDH-DSS-DES-CBC-SHA`	Extreme example.

Example for GSK:

`CIPHER-SUITE=090306`	Use DES and SHA1 with export key lengths, or RC4 and MD5 with export key lengths, or RC2 and MD5 with export key lengths.

For more information see:

- **OpenSSL**
  http://www.openssl.org/docs/apps/ciphers.html

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	■ **GSK** http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/apis/gsk_attribute_set_buffer.htm						
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. *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	*file name*	R		u	w		b
	File that contains the broker's private key (if not contained in KEY-STORE). (Example: *MyAppKey.pem*)						
KEY-PASSWD	*password* (A32)	R		u	w		b
	Password used to protect the private key. Unlocks *MyAppKey.pem*. Deprecated. See KEY-PASSWD-ENCRYTPED below.						
KEY-PASSWD-ENCRYPTED	*encrypted value* (A64)	R		u	w		b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Password used to protect the private key. Unlocks *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.						
KEY-STORE	*file name*	R		u	w		b
	SSL certificate; may contain the private key. (Example: *ExxAppCert.pem*)						
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*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 or TLS. It overrides the global value of trace level for all SSL or 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. 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 restarting the broker, use System Management Hub or ETBCMD. Trace levels 2, 3, and 4 should be used only when requested by Software AG support.						
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						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	be associated with the user ID that the broker kernel is running under. BS2000/OSD/Windows/UNIX Specify the file name of the CA certificate store. Examples: *EXXCACERT.PEM, C:\Certs\ExxCACert.pem*						
VERIFY-CLIENT	NO | YES	O	z	u	w		b
	YES Additional client certificate required. NO No client certificate required (default).						

DIV-specific Attributes

The DIV-specific attribute section begins with the keyword `DEFAULTS = DIV`. The attributes in this section are required if `PSTORE-TYPE = DIV` is specified.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
`DIV`	A511	R	z				
	The VSAM Persistent Store parameters, enclosed in double quotes (""). The value can span more than one line. See *Format Parameters* under *Managing the Broker Persistent Store* in the z/OS administration documentation for details of the parameters. In previous versions of EntireX, these parameters were read from the SYSIN DD during broker kernel startup.						

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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.						
TRACE-LEVEL	0-8	O	z	u	w	v	b
	Trace level for Adabas persistent store. It overrides the global value of trace level in the attribute file.						

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 attribute`BROKER-ID`)
- `NODE` (in Entire Net-Work-specific attribute `NODE`)
- `PORT` (in `PORT` (SSL) and `PORT` (TCP/IP))

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.

5 Concepts of Persistent Messaging

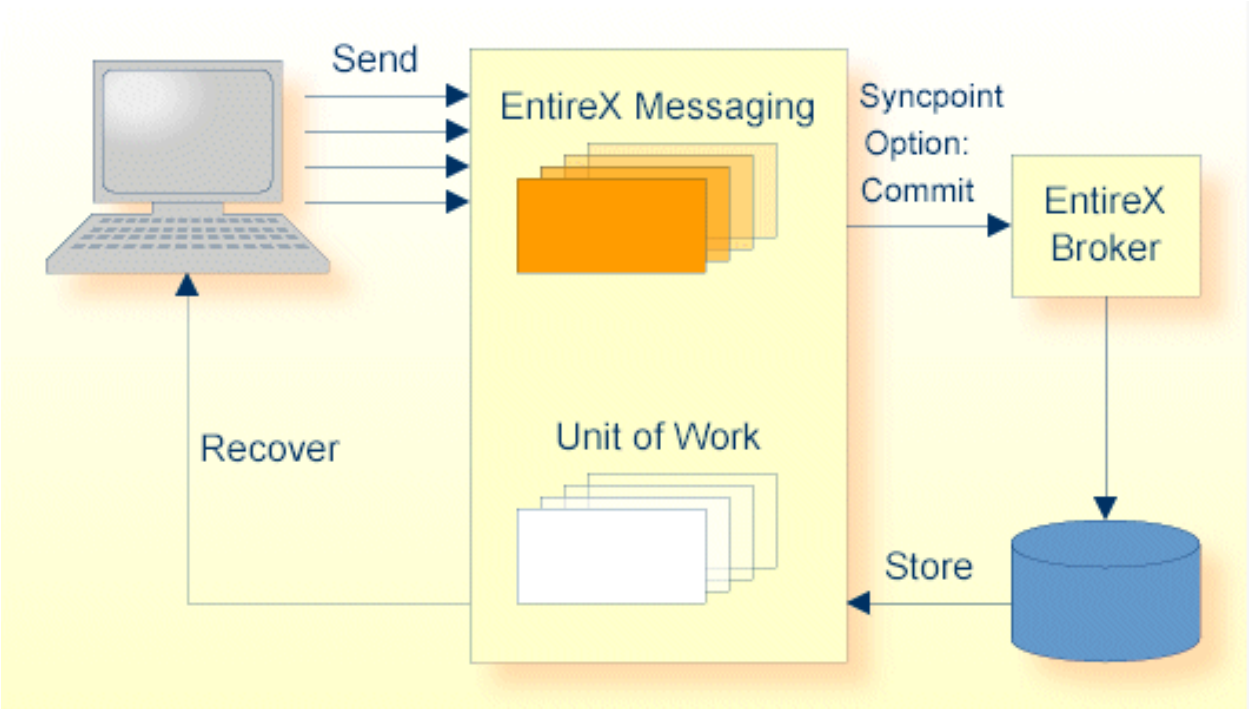
This chapter provides a brief introduction to the concepts of the persistent store and its role in EntireX for providing persistent messaging within the client/server model and also for publish-and-subscribe functionality. It covers the following topics:

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* under ***Concepts of Persistent Messaging*** 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.

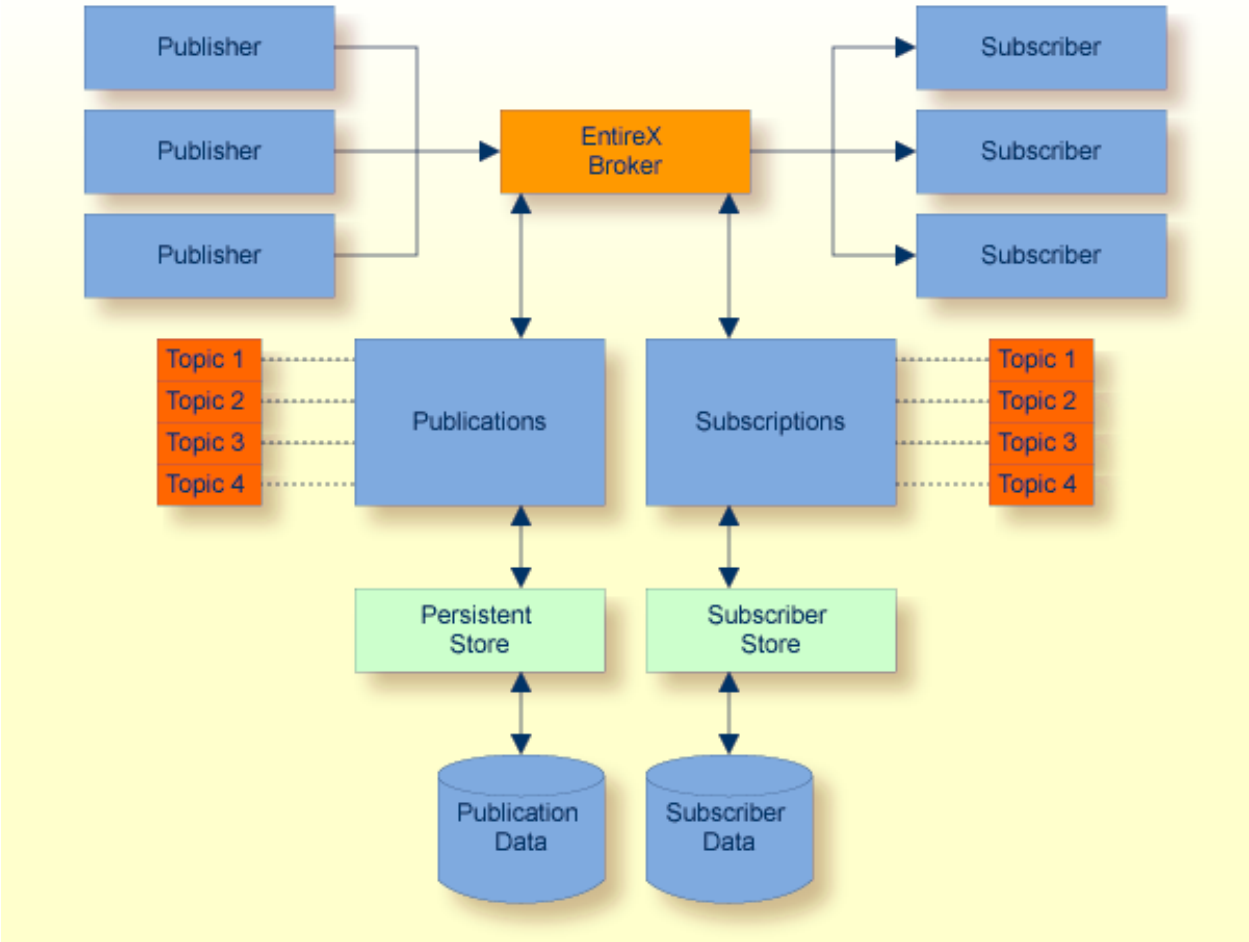
To support transaction control in a coordinated operation of distributed systems, EntireX can group logically related messages into "units of work" that are committed to the EntireX Broker for further transmission when complete. In case of failure on the server side, the receiving program can backout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

Publish-and-Subscribe Model: Persistent Behavior

EntireX provides persistent publish-and-subscribe behavior by writing information to disk in order to protect against system failures. This allows applications to be restarted without any loss of the following types of data:

- **Durable Subscription Information**
  This comprises a list of subscribers and the topics to which they have durably subscribed. This ensures that users only have to subscribe once to a topic; their persistent status remains after Broker is restarted. If the persistent store is used to maintain subscription status, you must define the `SUBSCRIPTION-EXPIRATION` options.
- **Publication Data**
  This data is also persisted if the administrator has defined this characteristic for the topic.

The diagram below shows the two types of publish-and-subscribe information which is written to the persistent store.


Publisher
Publisher
Publisher
EntireX Broker
Subscriber
Subscriber
Subscriber
Topic 1
Topic 2
Topic 3
Topic 4
Publications
Subscriptions
Topic 1
Topic 2
Topic 3
Topic 4
Persistent Store
Subscriber Store
Publication Data
Subscriber Data

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 and publish-and-subscribe data 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, 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.

Persistent Store Type	Description	Operating System	Notes
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* in the administration documentation.

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.

Publication

A publication is one or more messages forming an atomic unit and sent by a publisher to a topic. Subscribers are then able to receive publications committed after the time at which a subscriber first subscribes.

Durable Subscription

Subscribers inform EntireX of their intent to receive publications by issuing a `SUBSCRIBE` command and specifying the topic of interest. If the administrator has specified this topic to the Broker attribute file with a characteristic of `DURABLE`, users will be able to subscribe to the topic durably. This means that the user's subscription status remains after EntireX is restarted.

Publication Lifetime

A characteristic of the topic is the lifetime which publications will live and be available to subscribers. Once a publication has been received by all eligible subscribers, it will be removed automatically, even before its lifetime has been reached.

Subscription Expiration

Subscribers inform EntireX of their intent to receive publications by issuing a `SUBSCRIBE` command and specifying the topic of interest. If the administrator has specified this topic to the Broker attribute file with a characteristic of `DURABLE`, all user subscriptions to that topic will be durable. This means that the user's subscription status remains after EntireX is restarted.

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 and durable publish and subscribe - 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 or publications.

During the time the PSTORE is unavailable, there is no timeout processing for unit-of-work and publication 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* under *Broker Command and Information Services*.

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* under *Broker Command and Information Services*.

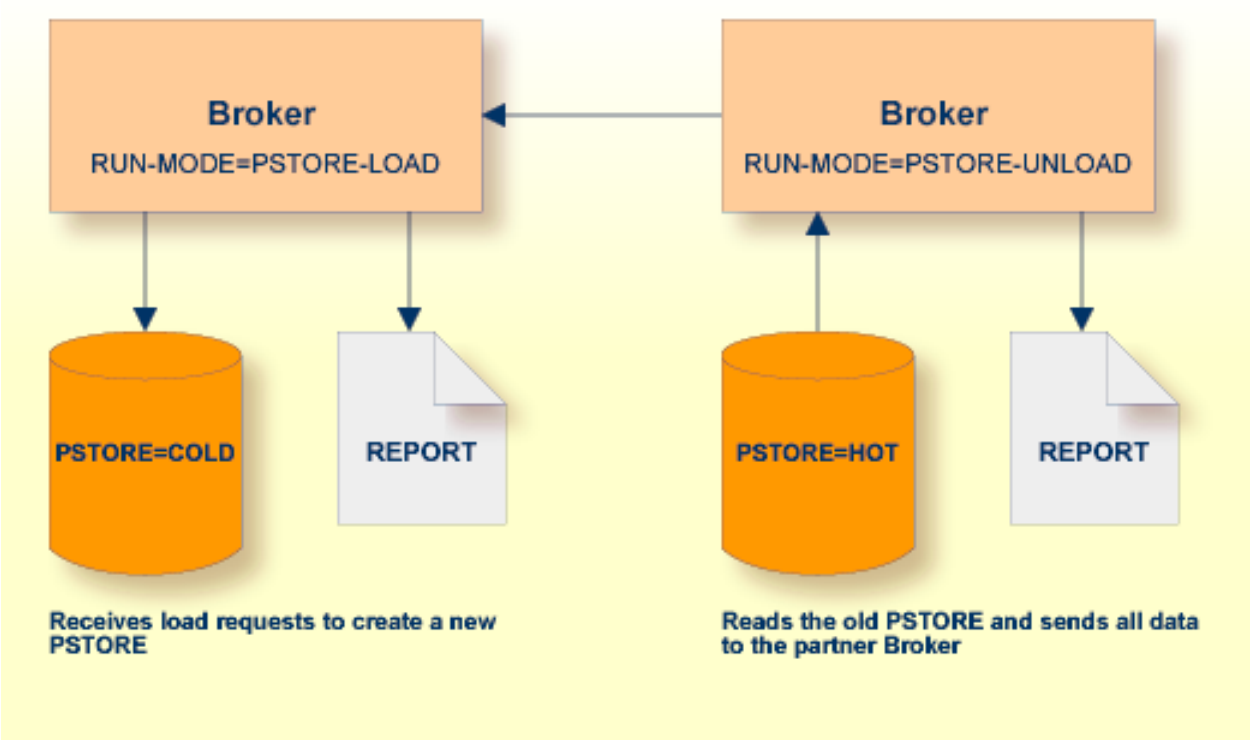
Migrating the Persistent Store

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 `RUNMODE` 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.



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* in the administration documentation 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

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/OSD

`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 8.0.0.00     PSTORE Report      2008-02-21 17:18:38    Page     1

Identifier        Elements  Total length  Record Type   Date       Time          ↩
Action
100000000D000016         5          1148   Conversation  2008-02-21 17:18:57.190  ↩
Created
100000000D000017         5          1148   Conversation  2008-02-21 17:18:57.654  ↩
Created
100000000D000018         5          1148   Conversation  2008-02-21 17:18:58.122  ↩
Created
100000000D000019         5          1148   Conversation  2008-02-21 17:18:58.590  ↩
Created
100000000D00001A         5          1148   Conversation  2008-02-21 17:18:59.054  ↩
```

```
Created
100000000D00001B        5          1148   Conversation  2008-02-21 17:18:59.518  ↩
Created
100000000D00001C        5          1148   Conversation  2008-02-21 17:18:59.982  ↩
Created
100000000D00001D        5          1148   Conversation  2008-02-21 17:19:00.538  ↩
Created
100000000D00001E        5          1148   Conversation  2008-02-21 17:19:01.002  ↩
Created
100000000C000001        0             0   Conversation  2008-02-21 17:19:30.676  ↩
Deleted
100000000C000002        0             0   Conversation  2008-02-21 17:19:31.675  ↩
Deleted
100000000C000003        0             0   Conversation  2008-02-21 17:19:32.675  ↩
Deleted
100000000C000004        0             0   Conversation  2008-02-21 17:19:33.675  ↩
Deleted
100000000C000005        0             0   Conversation  2008-02-21 17:19:34.675  ↩
Deleted
100000000C000006        0             0   Conversation  2008-02-21 17:19:35.675  ↩
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
  - `Publication`: a record containing a publication for a durable topic
  - `Subscription`: a record containing subscriber data (if `SUBSCRIBER-STORE` = `PSTORE`) is defined
- Date and time of the action
- `Action` describes the action of Broker. The following actions are currently supported:
  - `Created`: record is created
  - `Deleted`: record is deleted
  - `Loaded`: record is loaded (Broker instance with `RUN-MODE` = `PSTORE-LOAD`)
  - `Unloaded`: record is unloaded (Broker instance with `RUN-MODE` = `PSTORE-UNLOAD`)

Swapping out New Units of Work

The broker processes UOWs in memory. However, if a client produces a large number of UOWs and no server is available, or the server cannot handle all data, the number of UOWs in memory may increase and reach a critical limit.

To avoid an overload of UOWs in memory, EntireX Broker can swap out new conversations that containing UOWs (`STORE=BROKER`) and that have been accomplished by the client with an EOC. The data is persisted on PSTORE and there is no need to keep the data in memory unless a server wants to receive the data.

Activate the swap-out feature with the broker-specific attribute `SWAP-OUT-NEW-UOWS`. It is not activated by default. However, the swap-out feature can be configured per service to define a minimum portion of UOWs kept in memory. Use the service-specific attribute `MIN-UOW-CONVERSATIONS-IN-MEMORY` to define this portion.

6 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, 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.
UWTIME	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.

Keyword	Value	Description
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.
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. 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.

Keyword	Description
	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: 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.

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.

CID Implementation: Numeric Digits, Characters 0-9 and A-Z

In order to support unique conversation identifiers at Broker restart, there is an implementation of the CID which is alphanumeric and an internal identifier.

Note: The CID is a Broker-generated identifier for the conversation, and the application should not make any assumptions about either the content or format of all or any part of the CID field, or about any relationship between CIDs.

If any of the following three conditions exist, the all-numeric implementation of the CID field will be used in order to ensure compatibility:

- the Broker does not support any UOW processing;
- the application program is using `API_VERSION` 1 or 2 in its request;

  or

- the target service does not support UOWs.

Note: This level of compatibility may be removed at some point in the future.

Using Units of Work

UOW vs non-UOW Conversations

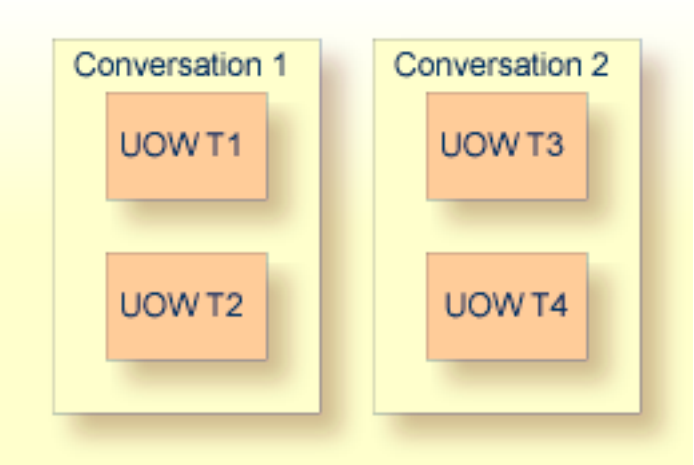
A Broker conversation will support either UOWs or messages, but not both. At the time the conversation is created, the Broker will determine which is to be supported.

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

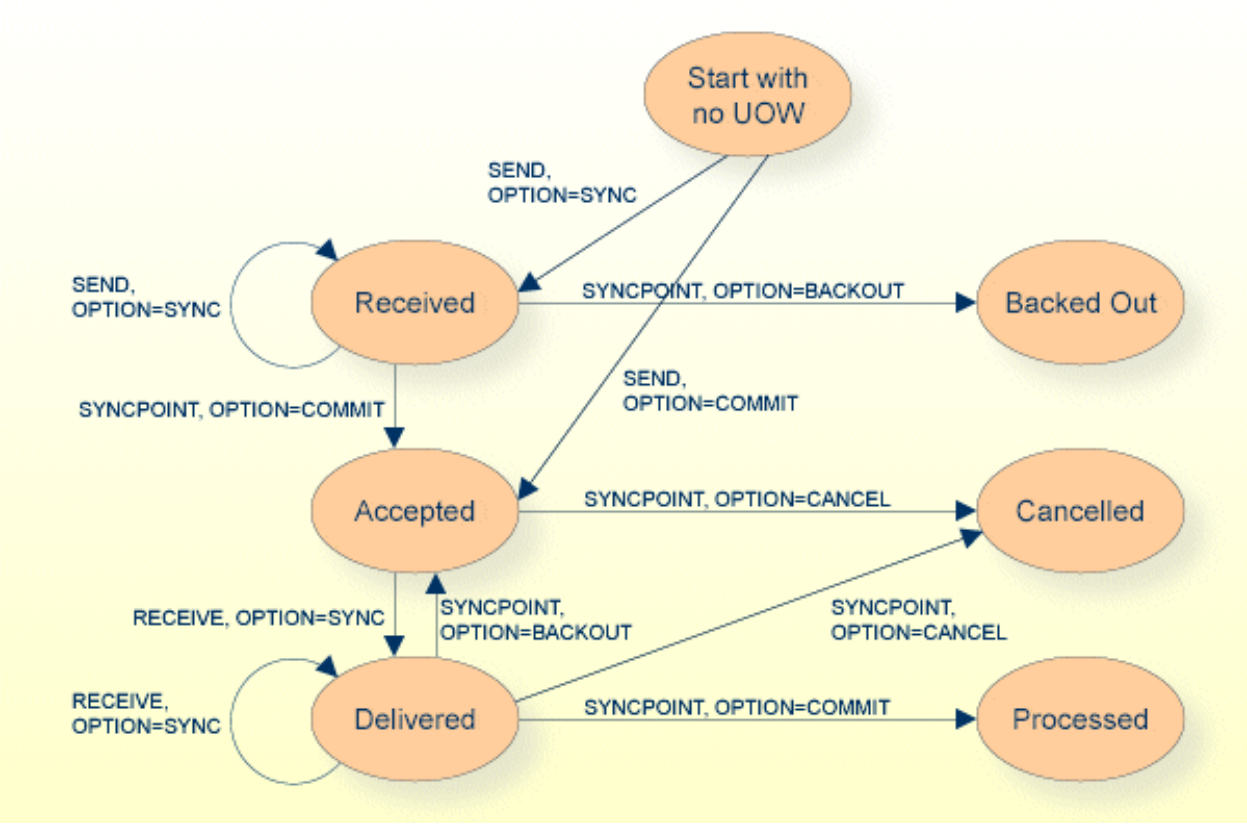
In order to use UOWs effectively, you need to understand

- 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 synchron-

ously 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?

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
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

UOW Status before Restart	Persistent UOW: YES | NO	Persistent Status: YES | NO	Behavior of UOW and Status	UOW Status after Restart *
DELIVERED	NO	NO	UOW not restored; Status 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.
UWCB	Receiver	UWCB = unit of work CB The UWCB represents the UOW.

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.

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

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.

7 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* in the ACI Programming documentation | *Broker ACI Functions* in the EntireX Broker ACI Programming documentation | *Error Messages and Codes*.

Initial UOW Status: NULL | Received

No.	Initial UOW Status	Action	Resulting UOW Status				Description
			PU&PS	PU&NPS	NPU&PS	NPU&NPS	
2	Received	Send	Received	Received	Received	Received	
3	Received	Commit	Accepted	Accepted	Accepted	Accepted	
4	Received	ReStart	BackedOut	NULL	Discarded	NULL	
5	Received	BackOut	BackedOut	NULL	BackedOut	NULL	
6	Received	TimeOut	BackedOut	NULL	BackedOut	NULL	R6: This action can only be a conversation timeout since a UOW only exists once it is committed.
7	Received	Delete	Received	Received	Received	Received	
8	Received	Cancel	Received	Received	Received	Received	
9	Received	Receive	Received	Received	Received	Received	

Initial UOW Status: Accepted | Delivered

No.	Initial UOW Status	Action	Resulting UOW Status				Description
			PU&PS	PU&NPS	NPU&PS	NPU&NPS	
10	Accepted	Receive	Delivered	Delivered	Delivered	Delivered	
11	Accepted	Timeout	Timedout	NULL	Timedout	NULL	
12	Accepted	Restart	Accepted	Accepted	Discarded	NULL	
13	Accepted	Cancel	Cancelled	NULL	Cancelled	NULL	
14	Accepted	Delete	Accepted	Accepted	Accepted	Accepted	
15	Accepted	BackOut	Accepted	Accepted	Accepted	Accepted	
16	Accepted	Send	Accepted	Accepted	Accepted	Accepted	
17	Accepted	Commit	Accepted	Accepted	Accepted	Accepted	
18	Delivered	Receive	Delivered	Delivered	Delivered	Delivered	
19	Delivered	Commit	Processed	NULL	Processed	NULL	
20	Delivered	Cancel	Cancelled	NULL	Cancelled	NULL	R20: Cancel can only be issued by receiver of the UOW
21	Delivered	BackOut	Accepted	Accepted	Accepted	Accepted	
22	Delivered	TimeOut	Timedout	NULL	NULL	NULL	
23	Delivered	Restart	Accepted	Accepted	Discarded	NULL	
24	Delivered	Delete	Delivered	Delivered	Delivered	Delivered	
26	Delivered	Send	Delivered	Delivered	Delivered	Delivered	

Initial UOW Status: Processed | Timedout

No.	Initial UOW Status	Action	Resulting UOW Status				Description
			PU&PS	PU&NPS	NPU&PS	NPU&NPS	
27	Processed	Delete	NULL	N/A	NULL	N/A	Processed is a STABLE UOW status:
28	Processed	Timeout	NULL	NULL	NULL	N/A	All actions and transitions refer to the status of a UOW.
29	Processed	Restart	Processed	N/A	Processed	N/A	
30	Processed	Backout	Processed	N/A	Processed	N/A	
31	Processed	Cancel	Processed	N/A	Processed	N/A	
32	Processed	Commit	Processed	N/A	Processed	N/A	
33	Processed	Receive	Processed	N/A	Processed	N/A	
34	Processed	Send	Processed	N/A	Processed	N/A	
35	Timedout	Restart	Timeout	N/A	Timeout	N/A	Timedout is a STABLE UOW status:
36	Timedout	Delete	NULL	N/A	NULL	N/A	All actions and transitions refer to the status of a UOW.
37	Timedout	Timeout	NULL	N/A	NULL	N/A	
38	Timedout	Send	Timedout	N/A	Timedout	N/A	
39	Timedout	Receive	Timedout	N/A	Timedout	N/A	
40	Timedout	Commit	Timedout	N/A	Timedout	N/A	
41	Timedout	Backout	Timedout	N/A	Timedout	N/A	
42	Timedout	Cancel	Timedout	N/A	Timedout	N/A	

Initial UOW Status: Cancelled | Discarded | Backedout

No.	Initial UOW Status	Action	Resulting UOW Status				Description
			PU&PS	PU&NPS	NPU&PS	NPU&NPS	
43	Cancelled	Delete	NULL	N/A	NULL	N/A	Cancelled is a STABLE UOW status:
44	Cancelled	Restart	Cancelled	N/A	Cancelled	N/A	All actions and transitions refer to the status of a UOW.
45	Cancelled	TimeOut	NULL	N/A	NULL	N/A	
46	Cancelled	Send	Cancelled	N/A	Cancelled	N/A	
47	Cancelled	Receive	Cancelled	N/A	Cancelled	N/A	
48	Cancelled	Commit	Cancelled	N/A	Cancelled	N/A	
49	Cancelled	Backout	Cancelled	N/A	Cancelled	N/A	
50	Cancelled	Cancel	Cancelled	N/A	Cancelled	N/A	
51	Discarded	Delete	N/A	N/A	NULL	N/A	Discarded is a STABLE UOW status:
52	Discarded	TimeOut	N/A	N/A	NULL	N/A	All actions and transitions refer to the status of a UOW.
53	Discarded	Restart	N/A	N/A	Discarded	N/A	
54	Discarded	Cancel	N/A	N/A	Discarded	N/A	
55	Discarded	Send	N/A	N/A	Discarded	N/A	
56	Discarded	Receive	N/A	N/A	Discarded	N/A	
57	Discarded	Commit	N/A	N/A	Discarded	N/A	
58	Discarded	Backout	N/A	N/A	Discarded	N/A	
59	BackedOut	TimeOut	NULL	N/A	NULL	N/A	BackedOut is a STABLE UOW status:
60	BackedOut	Cancel	BackedOut	N/A	BackedOut	N/A	All actions and transitions refer to the status of a UOW
61	BackedOut	Restart	BackedOut	N/A	BackedOut	N/A	
62	BackedOut	Send	BackedOut	N/A	BackedOut	N/A	
63	BackedOut	Receive	BackedOut	N/A	BackedOut	N/A	
64	BackedOut	Commit	BackedOut	N/A	BackedOut	N/A	
65	BackedOut	Delete	NULL	N/A	NULL	N/A	
66	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

8 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 Applications: Client and Server* | *Publish and Subscribe* 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

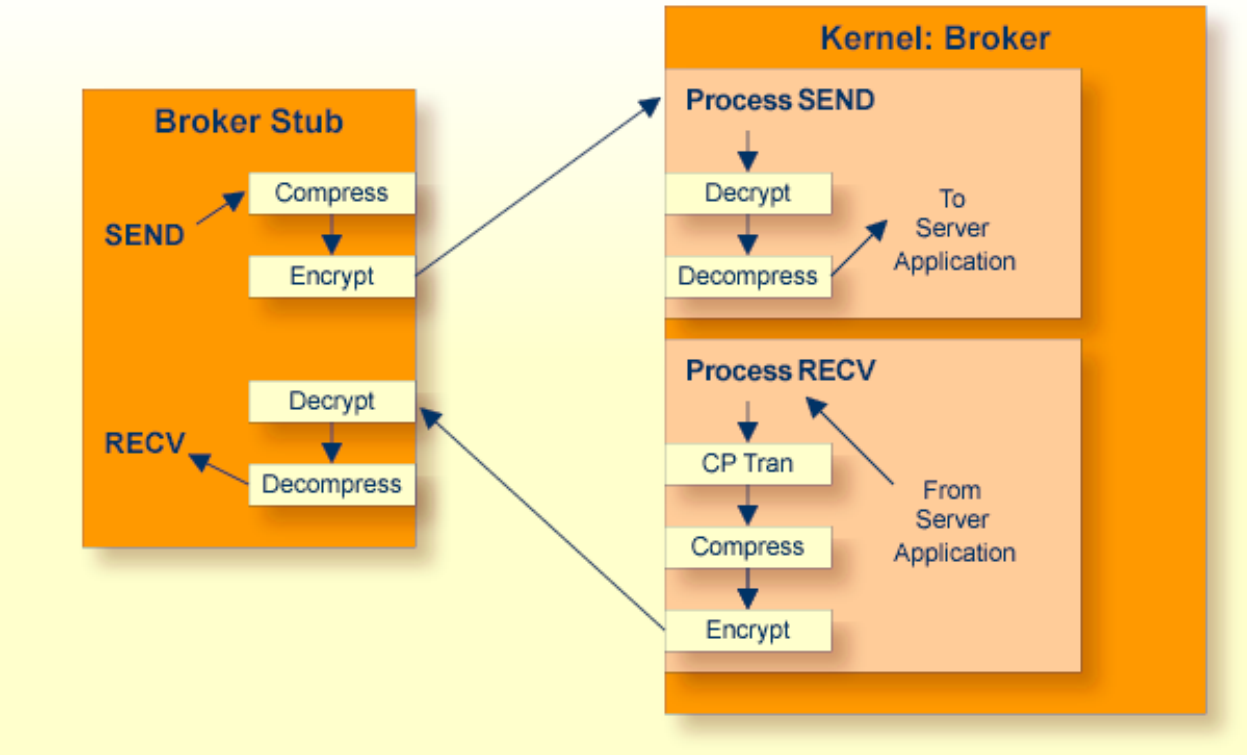
Compression of the data is implemented by the following components of EntireX:

Components	Description	
Broker control block	The Broker control block (ETBCB) contains a field that is used to set the compression level. This field determines for any `SEND/RECEIVE` transmission whether the data buffer will be compressed/decompressed. Possible values:	
	0 - 9	0 = no compression, 9 = maximum compression/decompression
	N	Default. No compression.

Components	Description
	Y Compression level 6 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. **Note:** See also ACI control block field `COMPRESSLEVEL`.
Stubs: Broker stub and Java stub	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 `SEND` or `RECEIVE` command. If the application issues a `SEND`, the stub compresses the data buffer before transmission of the data. If the application issues a `RECEIVE`, the stub decompresses the data buffer after reception of the data. **Note:** The compression level is used to control compression only between the application and the Broker kernel.
Broker kernel	When a client or server application `SEND`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 `RECEIVE` 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.

Sequencing Summary

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



Sample Programs

convClt and convSrv

Sample programs `convClt` and `convSrv` in directory *examples/ACI/conversational/C* can be used as an example of performing compression/decompression. Using the `-rn` option will cause compression to be used at level `<n>`.

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

```
convSrv -7 -r4
```

  - `-r4`: This will cause a compression/decompression level of 4 to be used on all transmissions between the server and the Broker.
  - `-7`: The -7 that is needed as compression/decompression is only supported at Version 7 or above.
- `convClt` can be instructed to use compression/decompression by specifying, for example:

```
convClt -7 -r2
```

- `-r2`: This will cause a compression/decompression level of 2 to be used on all transmissions between the client and the Broker.
- `-7`: The -7 that is needed as compression/decompression is only supported at Version 7 or above.

Option -g<filename>convClt and convSrv

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 extracted from a pre-existing file, using the two arguments from above.

```
convClt -7 -r2 -gmyfile1.txt
```

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

```
convSrv -7 -r4 -gmyfile2.txt
```

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

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	UNIX and Windows: A14 Timestamp in "YYYYMMDDHHMMSS" format z/OS: Timestamp	UNIX and Windows: The time this record was written to the accounting file in YYYYMMDDHHMMSS format z/OS: SMF timestamp.
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 8.1.2.00
Platform of Operation	1	A32	Platform where EntireX is running.
EntireX Start Time	1	A14 Timestamp in "YYYYMMDDHHMMSS" format	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.

Field Name	Accounting Version	Type of Field	Description
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 = WebSphere 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 = WebSphere 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.
Server Received Bytes	1	I4	Number of bytes received by server.
Server Sent Messages	1	I4	Number of messages sent by server.

Field Name	Accounting Version	Type of Field	Description
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	A14 Timestamp in "YYYYMMDDHHMMSS" format	Time conversation began in YYYYMMDDHHMMSS format.
Conversation End Time	1	A14 Timestamp in "YYYYMMDDHHMMSS" format	Time 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* in the ACI Programming documentation.
Client Credentials Type	2	I1	Mechanism by which authentication is performed for client.
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.

Field Name	Accounting Version	Type of Field	Description
Server Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field *APPLICATION-NAME* in the ACI Programming documentation.
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

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

For Broker and for Broker Services, the ACCOUNTING attribute/parameter 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

Broker Services Parameters

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 and Broker Services 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 EXX951.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 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* in the administration documentation. 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* in the administration documentation. 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.

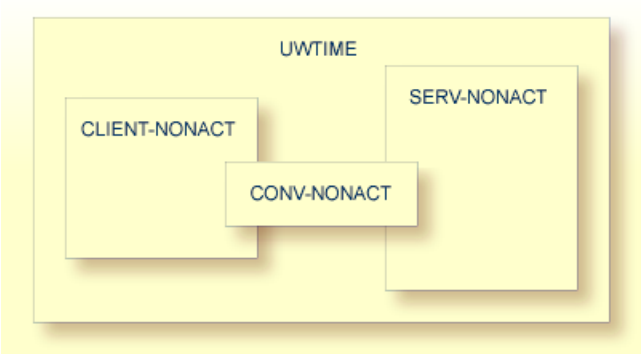
Relationship between Timeout Values

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.

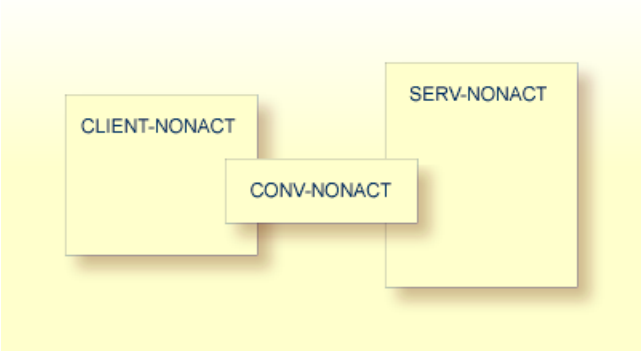
  Note: The link to the consumer is lost only for the first UOW in a conversation when the status changes to `ACCEPTED`; with subsequent UOWs, the link is not lost.

- 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.

11 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 EXB951.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.
```