webMethods EntireX

Administration under z/OS

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods EntireX Version 10.7 and all subsequent releases.

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

Document ID: EXX-ADMIN-107-20220422ZOS

Table of Contents

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 Setting up Broker Instances

This chapter contains information on setting up the Broker under z/OS. It assumes that you have completed the relevant steps described under *Installing EntireX under z/OS*.

Setting up TCP/IP Transport

The recommended way to set up the TCP/IP communicator is to define `PORT=nnnn` and optionally `HOST=x.x.x.x|hostname` and `STACK-NAME=stackname` under *TCP/IP-specific Attributes*.

However, if no port number is specified in the broker attribute file, the EntireX Broker kernel uses `getservbyname` to determine the TCP/IP port on which it will listen for incoming connections. The specified name is the value of `BROKER-ID` in the attribute file. An entry for this value must be made in the local machine's */etc/services* file. Example:

```
ETBnnn yyyyy/tcp # local host
```

where *etbnnn* is the `BROKER-ID` and *yyyyy* is the intended port number. This is the same place from which local Broker stubs will obtain the port information. If `getservbyname` fails, then a default port number of 1971 will be used. This is the same default port number that the stubs use.

Running Broker with SSL/TLS Transport

Introduction

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

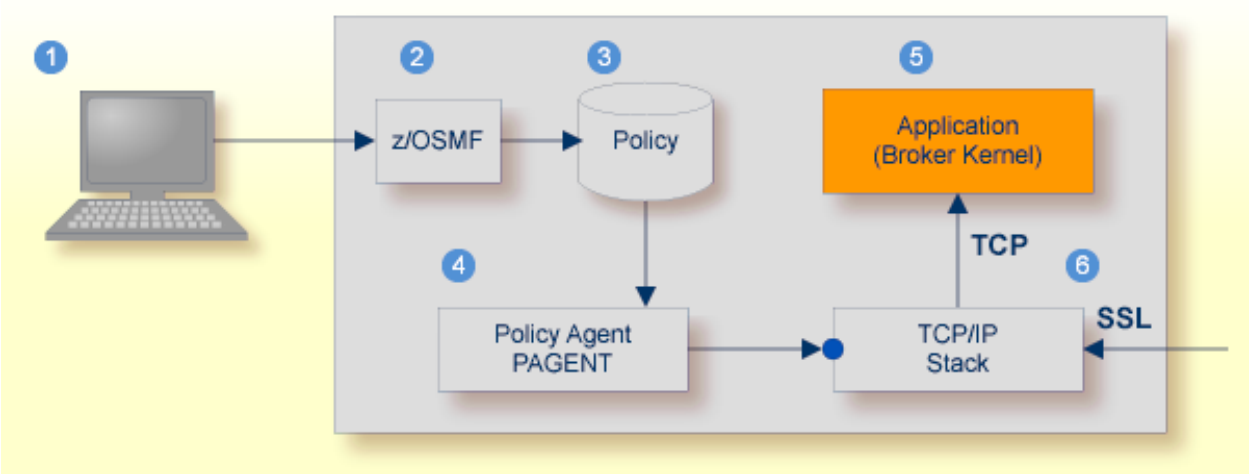
SSL delivered on a z/OS mainframe will typically use the Resource Access Control Facility (RACF) as the certificate authority (CA). Certificates managed by RACF can only be accessed through the RACF keyring container. A keyring is a collection of certificates that identify a networking trust relationship (also called a trust policy). In an SSL client/server network environment, entities identify themselves using digital certificates called through a keyring. Server applications on z/OS that wish to establish network connections to other entities can use keyrings and their certificate contents to determine the trustworthiness of the client or peer entity. Note that certificates can belong to more than one keyring, and you can assign different users to the same keyring. Because

of the way RACF internally references certificates, they must be uniquely identifiable by owner and label, and also unique by serial number plus data set name (DSN).

For establishing an SSL connection on z/OS, IBM's Application Transparent Transport Layer Security (AT-TLS) can be used, where the establishment of the SSL connection is pushed down the stack into the TCP layer.

Using IBM's Application Transparent Transport Layer Security (AT-TLS)

Configure the AT-TLS rules for the policy agent (`PAGENT`) 4 using an appropriate client 1 and the z/OS Management Facility (z/OSMF) 2. Together with SSL parameters (to provide certificates stored in z/OS as RACF keyrings) define AT-TLS rules, for example by using the application 5 job name and local TCP port number. If the rules match, the TCP connection is turned into an SSL connection 6. Refer to your IBM documentation for more information, for example the IBM Redbook *Communications Server for z/OS VxRy TCP/IP Implementation Volume 4: Security and Policy-Based Networking*.



1. Client to interact with z/OS Management Facility (z/OSMF).
2. AT-TLS rules are defined with z/OSMF policy management.
3. Policy Repository with AT-TLS rules stored as z/OS files.
4. Policy Agent, MVS task `PAGENT`, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.
5. Application using TCP connection.
6. If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

1. The client (1) may vary per operating system, for example a Web browser for z/OS 2.1.
2. z/OSMF (2) includes other administration and management tasks in addition to policy management.
3. Policy Management (3) includes other rules, such as IP filtering, network address translation etc.

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Modify broker-specific attributes. Configure the Broker to use TCP:

```
DEFAULTS = BROKER
  ...
  TRANSPORT = TCP

DEFAULTS = TCP
  PORT = 1958
```

See also ***Setting up TCP/IP Transport***.

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

4 Make sure the SSL clients connecting to the broker are prepared for SSL connections as well. See *Using SSL/TLS with EntireX Components*.

Migration from Broker's Direct SSL/TLS Support to AT-TLS

- Example

Steps

To migrate to AT-TLS

1 Migrate Broker SSL port to TCP.

Move the `PORT=value` line from the `DEFAULTS=SSL` section to the `DEFAULTS=TCP` section in the Broker attribute file to establish it as TCP/IP server port to be controlled by the Policy Agent `PAGENT` and TCP stack.

After this step you can delete the `DEFAULTS=SSL` section in the Broker attribute file.

2 Migrate Broker SSL attributes `KEY-LABEL` and `TRUST-STORE`.

Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

Part of this configuration is to provide the value of the Broker SSL attributes:

- `KEY-LABEL` as value of parameter `CertificateLabel` in the policy statement `TTLSConnectionAdvancedParms`, for example `CertificateLabel ExxAppCert`.
- `TRUST-STORE` as value of parameter `Keyring` in the policy statement `TTLSKeyringParms`, for example `Keyring EXX/EXXRING`.

z/OSMF Considerations

General steps to perform in the Network Configuration Assistant of the z/OS Management Facility:

1. Create a new Traffic Descriptor and specify
   - the local port for an SSL server (Broker) or remote port for an SSL client (EntireX client or server)
   - the TCP connect direction (inbound for the Broker and outbound for an EntireX client or server)
   - the AT-TLS handshake role (server for the Broker or client for EntireX client or server)

   You can specify the jobname to define the connection to the Broker. You also need to define the existing RACF keyring (format user ID/keyring) or OMVS keystore.
2. Expand or create a new Requirement Map to map the new Traffic Descriptor to an existing security level.

Note: The client and server need to use the same security protocol.

3. Create a new rule for the specific TCP/IP stack based on the Requirement Map. Install and activate the rule, using the PAGENT.

For more information on z/OSMF, see your IBM documentation.

Example

The following is an excerpt of a Policy Agent, MVS task `PAGENT` configuration file defining a Broker TCP/IP port as secure SSL port as outcome of this configuration:

```
TTLSRule                          ConnRule01~33
    {
      LocalAddr                       ALL
      RemoteAddr                      ALL
      LocalPortRangeRef               portR1
      RemotePortRangeRef              portR2
      Jobname                         <job_name> e.g. ETBNUC
      Direction                       Inbound
      Priority                        223
      TTLSConnectionActionRef         cAct1
      TTLSEnvironmentActionRef        eAct1
      TTLSGroupActionRef              gAct1
    }
    TTLSConnectionAction              cAct1
    {
      HandshakeRole                   Server
      TTLSCipherParmsRef              cipher1
      TTLSConnectionAdvancedParmsRef  cAdv1
      CtraceClearText                 Off
      Trace                           6
    }
    TTLSEnvironmentAction             eAct1
    {
      HandshakeRole                   Server
      EnvironmentUserInstance         0
      TTLSKeyringParmsRef             keyR1
    }
    TTLSGroupAction                   gAct1
    {
      TTLSEnabled                     On
      Trace                           6
    }
    TTLSConnectionAdvancedParms       cAdv1
    {
      CertificateLabel                <certificate_label> e.g. ExxAppCert
      SecondaryMap                    Off
      SSLv3                           Off
      TLSv1                           On
```

```
    TLSv1.1                         On
    TLSv1.2                         On
  }
  TTLSKeyringParms                  keyR1
  {
    Keyring                         <user_id / keyring> e.g. EXX/EXXRING
  }
  PortRange                         portR1
  {
    Port                            <port_number> e.g. 1958
  }
  PortRange                         portR2
  {
    Port                            1024-65535
  }
  CipherParms                       cipher1
  {
    V3CipherSuites                  TLS_RSA_WITH_AES_256_GCM_SHA384
    ...
    V3CipherSuites                  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  }
```

In this example, EntireX Broker is configured to establish TCP/IP port 1958 and the Policy Agent gets instructions to make 1958 an SSL server port.

Therefore, section `TTLSRule` needs `LocalPortRangeRef` set to port 1958 and `RemotePortRangeRef` set to ports 1024-65535. The jobname of the Broker is needed, and the `Direction` of an SSL server port is `Inbound`.

The `TTLSRule` also refers to the following sections:

- `TTLSConnectionActionRef`
  defines the `HandshakeRole Server` and refers to cipher parameters and advanced parameters for the connection
- `TTLSEnvironmentActionRef`
  defines the `HandshakeRole Server` and refers to the `Keyring` parameter
- `TTLSGroupActionRef`
  defines `TTLSEnabled On`

Advanced parameters for the connection define the sample `CertificateLabel` value `ExxAppCert` and the protocols to be used or not to be used.

The `Keyring` parameter specifies the RACF keyring using the following format: `USER-ID / KEY-RING-NAME`.

Setting up Entire Net-Work/Adabas SVC Transport

To set up EntireX Net-Work communication mechanism

1 Ensure that all load libraries in the Broker kernel steplib are APF-authorized.

2 Ensure that appropriate values are supplied in the Broker attribute file section `DEFAULTS=NET`, paying particular attention to the IUBL parameter - which specifies the maximum send/receive buffer length that can be sent between an application and Broker kernel within a single request - and NABS, which governs the total amount of storage available concurrently for all users communicating over this transport mechanism. See *Adabas SVC/Entire Net-Work-specific Attributes*.

3 Ensure that communication with the EntireX Broker is possible by running the installation verification programs using transport type NET. See *Sample Programs for Client (`BCOC`) and Server (`BCOS`)* in the z/OS Installation documentation.

Starting and Stopping the Broker

To start the Broker

1 Create a user ID for the started task or job where your Broker kernel is going to run.

2 If you are using a started task, ensure that the user ID is defined in the list of user IDs for started tasks and that the sample Broker kernel JCL is modified appropriately to create a PROC.

3 Start the Broker kernel either from the Broker kernel job (JCL) or started task (PROC).

To stop the Broker

- Issue the operator command `P <JOBNAME>`

  Or:

  Execute the `ETBCMD` utility using the example syntax below:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-bhost:port:TCP ',
//       '-cSHUTDOWN -dBROKER -xuid -ypwd')
```

See *Operator Commands* for a full list and also ***Broker Command-line Utilities***.

Dual Command Log Files

Command logging is a feature to assist in debugging Broker ACI applications. A command in this context represents one user request sent to the Broker and the related response of Broker.

Broker uses two command log files, enabling data to be written to one of the files while the other is being copied for archival purposes. Two file names must be specified for the dual command logs. At startup, Broker initializes both files and keeps the first open for printing command log data. Broker kernel switches to the other command log when the first file becomes full - or when the size of the open file reaches the value optionally specified by `CMDLOG-FILE-SIZE` (specified in KB).

Note: It is always advisable to copy the contents of a full command log file before Broker fills the subsequent command log file. Otherwise, the information in the first file (full and closed) will be overwritten.

The file requirements are two equally sized, physical sequential files defined with a record length of 121 bytes, i.e.
`DCB=(LRECL=121,RECFM=PS,BLKSIZE=nnnn)`. We recommend you allocate files with a single (primary) extent only. For example `SPACE=(CYL,(30,0))`. The minimum file size is approximately 3 cylinders of 3390 device.

Alternatively, the dual command log files can be allowed in USS HFS file system.

For more information, see ***Command Logging in EntireX***.

Tracing EntireX Broker

This section covers the following topics:

- Broker TRACE-LEVEL Attribute
- Attribute File Trace Setting
- Deferred Tracing
- Dynamically Switching On or Off the EntireX Broker Trace

- Flushing Trace Data to a GDG Data Set

Broker TRACE-LEVEL Attribute

The Broker TRACE-LEVEL attribute determines the level of tracing to be performed while Broker is running. The Broker has a master TRACE-LEVEL specified in the Broker section of the attribute file as well as several individual TRACE-LEVEL settings that are specified in the following sections of the attribute file.

Individual Settings	Specified in Attribute File Section	Note
Master trace level	DEFAULTS=BROKER	1,2
Persistent store trace level	DEFAULTS=ADABAS | CTREE | DIV	1
Conversion trace level	DEFAULTS=SERVICE; Trace option of the service-specific broker attribute CONVERSION.	
Security trace level	DEFAULTS=SECURITY	1
Transport trace level	DEFAULTS=NET | TCP | SSL	1
Application Monitoring trace level	DEFAULTS=APPLICATION-MONITORING	

Notes:

1. For temporary changes to the master or individual TRACE-LEVEL without restarting the Broker, use the Broker command-line utility ETBCMD.
2. For temporary changes to the master TRACE-LEVEL without restarting the broker, use operator command TRACE.

Attribute File Trace Setting

Trace Level	Description
0	No tracing. Default value.
1	Traces incoming requests, outgoing replies, and resource usage.
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.

Note: Trace levels 2 and above should be used only when requested by Software AG Support.

Deferred Tracing

It is not always convenient to run with TRACE-LEVEL defined, especially when higher trace levels are involved. Deferred tracing is triggered when a specific condition occurs, such as an ACI response code or a broker subtask abend. Such conditions cause the contents of the trace buffer to be written, showing trace information leading up the specified event. If the specified event does not occur, the Broker trace will contain only startup and shutdown information (equivalent to TRACE-LEVEL=0). Operating the trace in this mode requires the following additional attributes in the broker section of the attribute file. Values for TRBUFNUM and TRAP-ERROR are only examples.

Attribute	Value	Description
TRBUFNUM	3	Specifies the deferred trace buffer size = 3 * 64 K.
TRMODE	WRAP	Indicates trace is not written until an event occurs.
TRAP-ERROR	322	Assigns the event ACI response code 00780322 "PSI: UPDATE failed".

Dynamically Switching On or Off the EntireX Broker Trace

The following methods are available to switch on or off the EntireX Broker trace dynamically. You do not need to restart the broker for the changes to take effect.

- ETBCMD
  Run command utility ETBCMD with option -c TRACE-ON or - c TRACE-OFF. See ETBCMD.
- **Operator Command**
  Issue an operator command. See TRACE.

  See also *Deferred Tracing*.

Flushing Trace Data to a GDG Data Set

With broker-specific attributes TRMODE=WRAP and TRBUFNUM=*n*, Broker writes trace data to internal buffers instead of stderr (DD:SYSOUT). These buffers are used in round-robin mode and do not involve any I/O operation. If you need trace data for diagnostic purposes, use the operator command FLUSH to write the trace data from the internal buffers to a data set. A FLUSH command is performed automatically in case of error exceptions. The output data set is not readable for any other user when the broker is running. To avoid this problem, you can use a GDG (generation data group) data set as output data set. First you must allocate the GDG and define it to the broker. These preparatory steps are outlined below. The GDG name EXX.GDG is used in the examples.

Note: GDG is supported for deferred tracing only.

- Allocating a GDG
- Defining the GDG to Broker

Allocating a GDG

Define the GDG with IDCAMS. This definition is needed before working with the GDG data sets. You can use the following JCL to define a GDG. The `LIMIT` parameter is set to 16, but may contain other values according to your needs.

```
//IDCAMS   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DEFINE GENERATIONDATAGROUP -
  (NAME(EXX.GDG) -
  NOEMPTY -
  SCRATCH -
  LIMIT(16))
/*
```

Defining the GDG to Broker

The GDG data set as target for trace data can be managed without changes to the Broker JCL. However, the DD statement for such a GDG data set has to be defined as a Broker attribute in order to propagate the file characteristics to the runtime library of the IBM Language Environment.

```
TRACE-DD = "DSNAME=EXX.GDG,
            DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB),
            DISP=(NEW,CATLG,CATLG),
            SPACE=(CYL,(100,10)),
            STORCLAS=SMS"
```

See `TRACE-DD` under *Broker-specific Broker Attributes*.

Writing to the GDG Data Set

After successful broker initialization, a new data set of the GDG is allocated and opened. Based on the defined GDG name `EXX.GDG` in the sample JCL above, data set names `EXX.GDG.G0001V00`, `EXX.GDG.G0002V00`, `EXX.GDG.G0003V00` and so on will be allocated and written.

Use operator command `FLUSH` to write all trace data from internal buffers to the GDG data set. The data set will be closed at the end of the `FLUSH` processing, and the next GDG data set is allocated and opened. During broker shutdown, the GDG data set is filled with all available trace data and closed.

Protecting a Broker against Denial-of-Service Attacks

An optional feature of EntireX Broker is available to protect a broker running with `SECURITY=YES` against denial-of-service attacks. An application that is running with invalid user credentials will get a security response code. However, if the process is doing this in a processing loop, the whole system could be affected. If `PARTICIPANT-BLACKLIST` is set to `YES`, EntireX Broker maintains a blacklist to handle such “attacks”. If an application causes ten consecutive security class error codes within 30 seconds, the blacklist handler puts the participant on the blacklist. All subsequent requests from this participant are blocked until the `BLACKLIST-PENALTY-TIME` has elapsed.

Server Shutdown Use Case

Here is a scenario illustrating another use of this feature that is not security-related.

An RPC server is to be shut down immediately, using Broker Command and Information Services (CIS), and has no active request in the broker. The shutdown results in the `LOGOFF` of the server. The next request that the server receives will probably result in message 00020002 "User does not exist", which will cause the server to reinitialize itself. It was not possible to inform the server that shutdown was meant to be performed.

With the *blacklist*, this is now possible. As long as the blacklist is not switched off, when a server is shut down immediately using CIS and when there is no active request in the broker, a marker is set in the blacklist. When the next request is received, this marker results in message 00100050 "Shutdown IMMED required", which means that the server is always informed of the shutdown.

Setting the Time Zone for Broker

Broker obtains the time zone value by reading the environment variable `TZ`. If not specified, Broker retrieves the assignment of `TZ` from the configuration file */etc/profile*.

Check your /etc/profile for an appropriate setting of environment variable `TZ`. The `TZ` value should reflect the appropriate value for your location.

Remember that the new Daylight Saving Time rule according to the Energy Policy Act in the U.S. takes effect in 2007. If you live in an area affected by the new rule, you may append it to the `TZ` environment variable.

For example, `TZ=EST5EDT,M4.1.0,M10.5.0` is no longer valid. `TZ=EST5EDT,M3.2.0,M11.1.0` must be used instead.

If you don't want to change your /etc/profile, you may configure Broker's startup JCL to define environment variable `TZ`. Modify the `EXEC` statement thus:

```
//BROKER   EXEC PGM=ETBNUC,REGION=0M,TIME=1440,
//         PARM='ENVAR(''TZ=EST5EDT,M3.2.0,M11.1.0'')/'
```

The value of `TZ` should reflect the appropriate value for your location. Put the time zone value in quotation marks and parentheses, as outlined above. The rule for daylight saving time changes can be appended after the time zone value.

Achieving FIPS Compliance

The z/OS operating system can be configured to run EntireX Broker FIPS 140-2 compliant. This requires changes in the following system components:

- AT-TLS (Application Transparent Transport Layer Security)
- System SSL
- ICSF (z/OS Integrated Cryptographic Service Facility)

The TCP/IP port of Broker must run under the control of AT-TLS, and all applications using Broker need the Secure Socket Layer protocol (TLS/SSL) as transport. You do not need to change the attribute file of Broker.

The IBM support pages provide the document *Setting up AT-TLS for FIPS 140 mode.pdf*. It contains an overview of necessary system changes and describes the steps in detail.

3 Broker Attributes

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

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

Name and Location of Attribute File

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

Platform	File Name/Location
z/OS	Member *EXBATTR* in the EntireX Broker source library.

Attribute Syntax

Each entry in the attribute file has the format:

```
ATTRIBUTE-NAME=value
```

The following rules and restrictions apply:

- A line can contain multiple entries separated by commas.
- Attribute names can be entered in mixed upper and lowercase.
- Spaces between attribute names, values and separators are ignored.
- Spaces in the attribute names are not allowed.
- Commas and equal signs are not allowed in value notations.
- Lines starting with an asterisk (*) are treated as comment lines. Within a line, characters following an * or # sign are also treated as comments.
- The `CLASS` keyword must be the first keyword in a service definition.
- Multiple services can be included in a single service definition section. The attribute settings will apply to all services defined in the section.
- Attributes specified after the service definition (`CLASS`, `SERVER`, `SERVICE` *keywords*) overwrite the default characteristics for the service.
- Attribute values can contain variables of the form `${`*variable name*`}` or `$`*variable name*:
  - Due to variations in EBCDIC codepages, braces should only be used on ASCII (UNIX or Windows) platforms or EBCDIC platforms using the IBM-1047 (US) codepage.
  - The variable name can contain only alphanumeric characters and the underscore (_) character.
  - The first non-alphanumeric or underscore character terminates the variable name.
  - Under UNIX and Windows, the string `${`*variable name*`}` is replaced with the value of the corresponding environment variable.
  - On z/OS, variable values are read from a file defined by the DD name `ETBVARS`. The syntax of this file is the same as the attribute file.

- If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$variable name` will be used as the variable value.
- If you encounter problems with braces (and this is quite possible in a z/OS environment), we suggest you omit the braces.

Broker-specific Attributes

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
ABEND-LOOP-DETECTION	YES | NO	O	z	u	w	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	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. See also *Accounting in EntireX Broker* in the platform-specific Administration documentation.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
ACCOUNTING-VERSION	1 | 2 | 3 | 4 | 5	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. 5 Create accounting records in layout of version 5. This parameter applies when ACCOUNTING is activated.					
ACI-CONVERSION	YES | NO	O	z	u	w	b
	Determines the handling of ACI request and response strings of USTATUS. YES Convert ACI request and response strings with ICU. See *ICU Conversion* in the Internationalization documentation. NO Translate ACI request and response with internal translation table without support of national characters. See *Translation User Exit* in the Internationalization documentation. **Note:** This attribute was undocumented in EntireX versions prior to 10.3 and had default value NO. This meant that a translation user exit was used instead; this is no longer recommended.					
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	b
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See the separate Application Monitoring documentation.					
AUTOLOGON	YES | NO	O	z	u	w	b
	YES LOGON occurs automatically during the first SEND or REGISTER. NO The application has to issue a LOGON call.					
AUTOSTART	NO | YES	O		u	w	
	This attribute defines the autostart behavior of a broker. NO Broker is *not* started automatically with the next system start. YES Broker is restarted automatically with the next system start.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	Note: Prior to EntireX version 10.5 this was handled by the Broker Administration Service.					
BLACKLIST-PENALTY-TIME	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	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 Administration documentation.					
BROKER-ID	A32	R	z	u	w	b
	Identifies the broker to which the attribute file applies. The broker ID must be unique per machine. **Note:** The numerical section of the BROKER-ID is no longer used to determine the DBID in the EntireX Broker kernel with Entire Net-Work transport (NET). To determine the DBID, use attribute NODE in the DEFAULTS=NET section of the attribute file.					
CLIENT-NONACT	15M | *n* | *n*S | *n*M | *n*H	R	z	u	w	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	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	b
	Defines the maximum size of the file that the command log is written to, in kilobytes. The value must be 1024 or higher. The default value is 1024. When one command log file grows to this size, broker starts writing to the other file. For more details, see *Command Logging in EntireX*.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
CONTROL-INTERVAL	60S | *n* | *n*S | *n*M | *n*H	O	z	u	w	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	b
	Default number of conversations that are allocated for every service. UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM-CONVERSATION. *n* Number of conversations. This value can be overridden by specifying a CONV-LIMIT for the service. A value of 0 (zero) is invalid.					
DEFERRED	NO | YES	O	z	u	w	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	b
	YES An initial portion of memory is allocated at broker startup based on defined NUM-* attributes or internal default values if no NUM-* attributes have been defined. More memory is allocated without broker restart if there is a need to use more storage. Unused memory is deallocated. The upper limit of memory consumption can be defined by the attribute MAX-MEMORY. See *Dynamic Memory Management* under *Broker Resource Allocation* in the platform-independent Administration documentation. NO All memory is allocated at broker startup based on the calculation from the defined NUM-* attributes. Size of memory cannot be changed. This was the known behavior of EntireX 7.3 and earlier.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ NUM-CONV[ERSATION] ■ HEAP-SIZE ■ NUM-LONG[-BUFFER] ■ LONG-BUFFER-DEFAULT ■ NUM-SERVER ■ SERVER-DEFAULT ■ NUM-SERVICE-EXTENSION ■ SHORT-BUFFER-DEFAULT ■ NUM-SERVICE ■ NUM-CLIENT ■ NUM-SHORT[-BUFFER] ■ NUM-CMDLOG-FILTER ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-COMBUF ■ 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-QUEUE-DEPTH ■ WORKER-MIN ■ WORKER-START-DELAY ■ WORKER-NONACT The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See *Dynamic Worker Management*.					
ETBCOM	YES | NO	O				b
	Bundles the output of the various broker tasks in task ETBCOM.					
FORCE	NO | YES	O		u		

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	NO Use of ICU custom converters is not possible.					
IPV6	YES | NO	O	z	u	w	b
	YES Establish SSL and TCP/IP transport in IPv6 and IPv4 networks according to the TCP/IP stack configuration. NO Establish SSL and TCP/IP transport in IPv4 network only. This attribute applies to EntireX version 9.0 and above.					
LONG-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	b
	Number of long buffers to be allocated for each service. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-LONG-BUFFER. *n* Number of buffers. This value can be overridden by specifying a LONG-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.					
MAX-MEMORY	0 | *n* | *n*K | *n*M | *n*G | UNLIM	O	z	u	w	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	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	b
	Maximum number of messages in a unit of work.					
MAX-MSG	See MAX-MESSAGE-LENGTH.					
MAX-TRACE-FILES	4 | *n*	O		u	w	
	Defines the number of backup copies of the trace file ETB.LOG. Minimum number is 1; maximum is 999. A new trace file is allocated when the value for TRACE-FILE-SIZE is exceeded. These two attributes prevent a constantly growing ETB.LOG file. See *Trace File Handling* under UNIX | Windows.					
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.					
MAX-UOWS	0 | *n*	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	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	b
	Indicates if certain error message texts returned by the broker to its clients or written by the broker to its log file are to be in mixed case, uppercase, or lowercase. NONE No changes are made to message case. UPPER Messages are changed to uppercase. LOWER Messages are changed to lowercase.					
MUOW	See NUM-UOW.					
NEW-UOW-MESSAGES	YES | NO	O	z	u	w	b
	YES New UOW messages are allowed. NO New UOW messages are not allowed. This applies to UOW when using Persistence and should not be used for non-persistent UOWs. A usage example could be the following: The broker persistent store reaches capacity and the broker shuts down. You can set NEW-UOW-MESSAGES to NO to prevent new UOW messages from being added after a broker restart. This action allows only consumption (not production) of UOWs to occur after broker restart. After the persistent store capacity has been sufficiently reduced, the EntireX Broker administrator can issue a CIS command, see ALLOW-NEWUOWMSGS. This action allows new UOW messages to be sent to the broker. Reset attribute NEW-UOW-MESSAGES to YES, which permits new UOW messages to be produced in subsequent broker sessions.					
NUM-BLACKLIST-ENTRIES	256 | *n*	O	z	u	w	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 Administration documentation.					
NUM-CLIENT	*n*	R	z	u	w	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	b
	Maximum number of filters that can be specified simultaneously.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	Tip: We recommend you limit this value to the number of services that are being monitored. Minimum value is 1. A value of zero is invalid when the attribute CMDLOG is set to YES. See *Command Logging in EntireX* in the EntireX Broker documentation for more information.					
NUM-COMBUF	1024 | 1-999999	R	z	u	w	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	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. Do not set the values used in the calculation to UNLIM. **Note:** 1. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See ***Wildcard Service Definitions***.					
NUM-LONG-BUFFER or NUM-LONG	4096 | *n* | AUTO	R	z	u	w	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. Do not set the values used in the calculation to UNLIM. A value of 0 (zero) is invalid. 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.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	In *conversational* mode, the last message received is always kept until a new one is received. **Note:** 1. If a catch-all service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See ***Wildcard Service Definitions***.					
NUM-PARTICIPANT-EXTENSION	*n*	O	z	u	w	b
	Defines the number of participant extensions to link participants as clients and servers. *n* Number of participant extensions. *not specified* If this attribute is not set, the default value is calculated based on NUM-CLIENT and NUM-SERVER. A value of 0 (zero) is invalid.					
NUM-SERVER	*n* | AUTO	R	z	u	w	b
	Defines the number of servers that can offer services concurrently using the broker. This is *not* the number of services that can be registered to the broker (see NUM-SERVICE). *n* Number of servers. AUTO Uses the SERVER-DEFAULT and the service-specific SERVER-LIMIT values to calculate the number of servers. Do not set the values used in the calculation to UNLIM. **Note:** 1. Setting this value higher than the number of services allows the starting of server replicas that provide the same service. 2. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 3. See ***Wildcard Service Definitions***.					
NUM-SERVICE	*n*	R	z	u	w	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	b
	Defines the number of service extensions to link servers to services.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	n Number of service extensions. AUTO Uses the value specified or calculated for NUM-SERVER + NUM-CLIENT, plus an extra cushion. *not specified* If this attribute is not set, the default value is NUM-SERVER multiplied by NUM-SERVICE. The minimum value is NUM-SERVER. The maximum value is NUM-SERVER multiplied by NUM-SERVICE. Caution is recommended with this attribute: ■ Set this attribute only if the storage resources allocated for service extensions need to be restricted. ■ Note that the value *n* allows only the specified number of server instances of *n* to be used. ■ Value AUTO will calculate the number of allowed server instances from NUM-SERVER, which itself might be set to AUTO. In this case, this also considers the value of SERVER-DEFAULT and even the individual SERVER-LIMIT for each service definition.						
NUM-SHORT-BUFFER or NUM-SHORT	*n* | AUTO	R	z	u	w	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. Do not set the values used in the calculation 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 Definitions*.						

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
NUM-UOW	0 | *n*	O	z	u	w	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	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	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	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 Administration documentation.					
PARTNER-CLUSTER-ADDRESS	A32	R	z	u	w	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.					
PERCENTAGE-FOR-CONNECTION-SHORTAGE-MESSAGE	90 | 1-100	O	z	u	w	b
	Broker will issue a message if the defined percentage value of TCP/IP connections (available file descriptors) is exceeded. Default is 90 percent of the available file descriptors.					
POLL	YES | NO	O	z	u		
	In earlier EntireX versions, the maximum number of TCP/IP connections per communicator was limited; see ***Maximum TCP/IP Connections per Communicator***					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	for platform-specific list. With attribute POLL introduced in EntireX version 9.0, this restriction can be lifted under z/OS and UNIX. NO This setting is used to run the compatibility mode in Broker. The poll() system call is not used. The limitations described under ***Maximum TCP/IP Connections per Communicator*** apply. YES The poll() system call is used to lift the resource restrictions with select() in multiplexing file descriptor sets. **Note:** The maximum number of file descriptors per process is a hard limit that cannot be exceeded by POLL=YES. Setting this attribute to YES increases CPU consumption. POLL=YES is only useful if ■ you need more than the maximum number of TCP/IP connections per communicator, as described under ***Maximum TCP/IP Connections per Communicator***, and ■ this maximum number is less than the maximum number of file descriptors per process We recommend POLL=NO to reduce CPU consumption.					
POSTPONED-QUEUE	YES | NO	O	z	u	w	
	Enable or disable the creation of a postponed queue for Broker. YES Enable creation of a postponed queue. Define your postponed queue with service-specific attributes POSTPONE-ATTEMPTS and POSTPONE-DELAY. NO Disable creation of a postponed queue. See *Postponing Units of Work*.					
PSTORE	NO | HOT | COLD	O	z	u	w	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	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	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	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*. `CTREE` c-tree database. UNIX and Windows only. See ***c-tree-specific Attributes*** and *c-tree Database as Persistent Store* under UNIX | Windows in the UNIX | 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 | 5`	O	z	u	w	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. The DIV PSTORE requires `PSTORE-VERSION=4`. `PSTORE-VERSION=5` was added in EntireX version 10.1 to support 64-bit time values on z/OS, and unique message IDs on all platforms. See *Unique Message ID*. `PSTORE-VERSION=5` significantly improvement Adabas PSTORE performance on all platforms. We strongly recommend you use this version. **Caution:** ■ If you go back to `PSTORE-VERSION=2` after upgrading to `PSTORE-VERSION=3`, the broker will only process data previously created with version 2. No version 3 data will be accessible. ■ If you change the DIV PSTORE from version 3 to 4, perform a COLD restart for the change to take effect, or run `PSTORE UNLOAD/LOAD` first. ■ If you change to `PSTORE-VERSION=5`, perform a COLD restart for the change to take effect.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	Note: Persistent Stores with PSTORE-VERSION less than 5 will no longer be supported after EntireX version 10.7.					
RUN-MODE	STANDARD | STANDBY | PSTORE-LOAD | PSTORE-UNLOAD	O	z	u	w	b
	Determines the initial run mode of the broker. STANDARD Default value. Normal mode. STANDBY Deprecated. Supported for compatibility reasons. PSTORE-LOAD Deprecated. Broker will run as load broker to write Persistent Store data to a new persistent store. See also *Migrating the Persistent Store*. PSTORE-UNLOAD Deprecated. 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*. **Note:** RUN-MODE options PSTORE-LOAD and PSTORE-UNLOAD are deprecated and will not be supported in the next version of EntireX.					
SECURITY	NO | YES	O	z	u	w	b
	Determines whether EntireX Security is activated. NO EntireX Security is not activated. YES EntireX Security is activated. See *EntireX Security*.					
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	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	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.					

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

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

	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"`
	
	Note: If you specify TRACE-DD, you must also specify TRMODE=WRAP and a value for TRBUFNUM for the setting to take effect.					

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	NO If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will stop. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will stop. YES If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will retry every 10 seconds to initiate communications with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will retry every 10 seconds to rebuild the persistent data. Broker will reject any user requests until it is able to rebuild the persistent data.					
WORKER-MAX	32 | *n* (min. 1, max. 32)	O	z	u	w	b
	Maximum number of worker tasks the broker can use.					
WORKER-MIN	1 | *n* (min. 1, max. 32)	O	z	u	w	b
	Minimum number of worker tasks the broker can use.					
WORKER-NONACT	70S | *n* | *n*S | *n*M | *n*H	O	z	u	w	b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*S. *n*S Non-activity time in seconds (default 70, max. 2147483647). *n*M Non-activity time in in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). **Caution:** A value of 0 (zero) is invalid. If you set this value too low, additional overhead is required for starting and stopping worker tasks. The default and recommended value is 70S.					
WORKER-QUEUE-DEPTH	1 | *n* (min. 1)	O	z	u	w	b
	Number of unassigned user requests in the input queue before another worker task gets started. The default and recommended value is 1. A higher value will result in longer broker response times.					
WORKER-START-DELAY	*internal-value* | *n*	O	z	u	w	b
	n Delay is extended by *n* seconds. Delay after a successful worker task invocation before another worker task can be started to handle current incoming workload. This attribute is used to avoid the risk of recursive invocation of worker tasks, because starting a worker task itself causes workload increase.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	If no value is specified, an internal value calculated by the broker is used to optimize dynamic worker management. This calculated value is the maximum time required to start a worker task.					

Service-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	b
	YES Enable application monitoring for the specified services. NO Disable application monitoring for the specified services. See the separate Application Monitoring documentation.					
APPLICATION-MONITORING-NAME or APPMON-NAME	A100	O	z	u	w	b
	Specifies the application monitoring name. Used to set the value of the **ApplicationName** KPI. If omitted, the default value from the APPLICATION-MONITORING section is used. If this value is also not specified, the corresponding CLASS/SERVER/SERVICE names are used. See the separate Application Monitoring documentation.					
CLASS	A32 (case-sensitive)	R	z	u	w	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. The following rules apply: ■ Classes starting with any of the following are reserved for use by Software AG. Do not use these in applications you write: 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.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	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	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 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	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	A255 (SAGTCHA[,TRACE=*n*][,*OPTION*=*s*]| SAGTRPC[,TRACE=*n*][,*OPTION*=*s*]| *name*[,TRACE=*n*]| NO)	O	z	u	w	b
	Defines ICU conversion or SAGTRPC user exit for character conversion. See *Internationalization with EntireX*. SAGTCHA [(1)] Conversion using ICU Conversion for *ACI-based Programming*.					

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
DEFERRED	NO | YES	O	z	u	w	b
	NO Units of work cannot be sent to the service until it is available. YES Units of work can be sent to a service that is not up and registered. The units of work will be processed when the service becomes available.					
LOAD-BALANCING	YES | NO	O	z	u	w	b
	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	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	b
	Maximum number of messages in a UOW.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	b
	Maximum message size that can be sent to a service. This is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.					
MAX-MSG	See MAX-MESSAGE-LENGTH.					
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.					
MAX-UOWS	0 | *n*	O	z	u	w	b
	0 The service does not accept units of work, that is, 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	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.					
MUOW	See MAX-UOWS.					
NOTIFY-EOC	NO | YES	O	z	u	w	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. 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.					
POSTPONE-ATTEMPTS	0 | *n*	O	z	u	w	
	Defines the number of attempts putting a received unit of work (UOW) due to SYNCPOINT option CANCEL on the postponed queue for later processing. 0 All UOWs rejected by the receiver (SYNCPOINT option CANCEL) will be cancelled immediately. Attribute POSTPONE-DELAY is ignored. *n* Defines the number of postpone attempts that are performed instead of considering the UOW finished due to SYNCPOINT option CANCEL; the UOW will be moved to the postponed queue and the UOW status will be changed to POSTPONED. These UOWs will be delivered to the receiver when the time specified with POSTPONE-DELAY has elapsed. **Note:** Broker-specific attribute POSTPONED-QUEUE must be enabled (default) for this attribute to take effect. The default value is 0. See *Postponing Units of Work*.					
POSTPONE-DELAY	0 | *n* | *n*S | *n*M | *n*H	O	z	u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	The length of time a UOW is kept in status POSTPONED. 0 No postponed queue is created and attribute POSTPONE-ATTEMPTS is ignored. *n*S Number of seconds the UOW stays unreadable in the postponed queue with status POSTPONED (max. 2147483647). *n*M Number of minutes the UOW stays unreadable in the postponed queue with status POSTPONED (max. 35791394). *n*H Number of hours the UOW stays unreadable in the postponed queue with status POSTPONED (max. 596523). *n*D Number of days the UOW stays unreadable in the postponed queue with status POSTPONED (max. 24855). The status of the UOW will be changed from POSTPONED to ACCEPTED after elapsed POSTPONE-DELAY. This delay time does not affect the UOW-DATA-LIFETIME. The POSTPONE-DELAY must be less than UOW-STATUS-LIFETIME in order to make the UOW receivable again. **Note:** Broker-specific attribute POSTPONED-QUEUE must be enabled (default) for this attribute to take effect. The default is 0, that is, no postponed queue is created, but if a value is entered, the minimum delay is 30 seconds. Any value entered that is less than 30 seconds will be increased to this value. See *Postponing Units of Work*.					
SERVER	A32 (case-sensitive)	R	z	u	w	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	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	b
	Allows a number of servers especially for this service.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	n Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO in the Broker section of the attribute file. A value of 0 (zero) is invalid. If NUM-SERVER=AUTO is specified in the Broker section of the attribute file, SERVER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SERVER-LIMIT attribute must be suppressed entirely for the service so that the default (SERVER-DEFAULT) becomes active. **Note:** UNIX and Windows: This limit also includes any attach server you are using. Make sure you increase the number by one for each attach server you use.					
SERVER-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	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	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	b
	Allocates a number of short message buffers for the service.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-SHORT-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of short message buffers. If NUM-SHORT-BUFFER=AUTO is specified in the Broker section of the attribute file, SHORT-BUFFER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the service so that the default (SHORT-BUFFER-DEFAULT) becomes active.						
STORE	OFF | BROKER	O	z	u	w	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	NO | *name* (A255)	O	z	u	w	b
	Activates translation user exit for character conversion. NO If translation is not to be used - for example for binary payload (broker messages) - either omit the TRANSLATION attribute or specify TRANSLATION=NO. *name* Name of Translation User Exit. See also *Configuring Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation or *Writing Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. The CONVERSION attribute overrides the TRANSLATION attribute when defined for a service; that is, when TRANSLATION and CONVERSION are both defined, TRANSLATION will be ignored.						
UMSG	Alias for MAX-MESSAGES-IN-UOW.					
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	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).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	This attribute is ignored if PSTORE=NO is defined. If the unit of work (UOW) is inactive, that is, not processed within the time limit, it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block.					
UOW-MSGS	Alias for MAX-MESSAGES-IN-UOW.					
UOW-STATUS-LIFETIME	no value | *n*[S]|*n*M|*n*H|*n*D	O	z	u	w	b
	The value to be added to the UOW-DATA-LIFETIME lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself. *n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523). *n*D Number of days (max. 24855). The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.					
UWSTATP	0 | *n*	O	z	u	w	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-STATUS-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. This attribute is ignored if PSTORE=NO is defined. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.					
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.					
UWTIME	Alias for UOW-DATA-LIFETIME.					

Wildcard Service Definitions

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

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

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

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

Service Update Modes

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

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

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

OPTION Values for Conversion

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

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

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

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

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)
STOP	Signals an error on detecting a non-convertible or bad input character. This is the default behavior if no option is specified.	YES	YES	Write detailed conversion error message.	Write detailed conversion error message.

Codepage-specific Attributes

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

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

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

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

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

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

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	DEFAULTS=CODEPAGE * Broker Locale String Codepage Assignments ASCII=ISO8859 EUC_JP_LINUX=ibm-33722_P12A-1999 * Customer-written ICU converters CP1140=myebcdic CP0819=myascii For more examples, see *Bypassing Broker's Built-in Locale String Mapping* and also ***Additional Notes*** below.					

Additional Notes

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

Adabas SVC/Entire Net-Work-specific Attributes

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
ADASVC	*nnn*	R	z			
	Sets the Adabas SVC number for EntireX Broker access. The Adabas SVC is used to perform various internal functions, including communication between the caller program and EntireX Broker. Not supported on BS2000.					
EXTENDED-ACB-SUPPORT	NO | YES	O	z			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			b
	Determines whether DBID table entries can be overwritten. NO Overwrite of DBID table entries not permitted. YES Overwrite of DBID table entries permitted. This is required when the DBID table entry is not deleted after abnormal termination. **Caution:** Overwriting an existing entry prevents any further communication with the overwritten node. Use FORCE=YES only if you are absolutely sure that no target node with that DBID is active.					
IDTNAME	*idtname*(A8) | ADABAS5B	O				b
	If an ID table name is specified with the appropriate ADARUN parameter for Entire Net-Work, Adabas or Natural, the same name must be specified here.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	The ID table is used to perform various internal functions, including communication between the caller program and the EntireX Broker. Only supported under BS2000.					
IUBL	8000 | *n*	O	z			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			b
	For remote nodes accessed via Entire Net-Work, the attribute LOCAL specifies whether the target ID defined with the NODE attribute can be accessed only locally, or also remotely. NO DBID is *global* and can be accessed from remote nodes via Entire Net-Work. YES DBID is *local* and cannot be accessed from remote nodes via Entire Net-Work.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	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			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			b
	NCQE defines the number of command queue elements which are available for processing commands arriving at the broker kernel over Adabas SVC / Net-Work transport mechanism. Sufficient NCQE should be allocated to allow this transport mechanism to process multiple broker commands concurrently. Each command queue element requires 192 bytes, and the element is released when either the user (client or server) has received the results of the command, or if the command is timed out.					

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

Security-specific Attributes

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

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

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

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

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

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

	`LDAP-AUTHENTICATION-URL="ldaps://HostName[:PortNumber]"`
	
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Examples for TCP and SSL/TLS:
	
	`LDAP-AUTHENTICATION-URL="ldap://myhost.mydomain.com"`
	`LDAP-AUTHENTICATION-URL="ldaps://myhost.mydomain.com:636"`					

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

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

`LDAP-AUTH-DN`	*`authDN`*	O		u	w	
	For authenticated access to the LDAP server. Specifies the DN of the user. Default value: `cn=admin,dc=software-ag,dc=de` This attribute replaces parameter `authDN` in the *xds.ini* file of EntireX version 9.10 and below.					
`LDAP-AUTH-PASSWD-ENCRYPTED`	*`authPass`*	O		u	w	
	For authenticated access to the LDAP server. Specifies the encrypted value of the user password. Use program `etbnattr` to get the encrypted password: `etbnattr -x clear_text_password -echo_password_only` This writes the encrypted password to standard output. This attribute replaces parameter `authPass` in the *xds.ini* file of EntireX version 9.10 and below.					
`LDAP-AUTHORIZATION-RULE`	`A32`	O		u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	List of authorization rules. Multiple sets of rules can be defined, each set is limited to 32 chars. The maximum number of LDAP-AUTHORIZATION-RULE entries in the attribute file is 16. Applies only when using EntireX Security under UNIX or Windows and SECURITY-SYSTEM=*ldapUrl*. Authorization rules can be stored in an LDAP repository. When an authorization call occurs, EntireX Security uses the values of this parameter and AUTHORIZATION-DEFAULT to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.						
LDAP-BASE-DN	*baseDN*	O		u	w	
	Specifies the base distinguished name of the directory object that is the root of all objects for authorization rules. Default value: `dc=software-ag,dc=de` This attribute replaces parameter baseDN in the *xds.ini* file of EntireX version 9.10 and below.					
LDAP-PERSON-BASE-BINDDN	*ldapDn*	O		u	w	
	Used with LDAP authentication to specify the distinguished name where authentication information is stored. This value is prefixed with the user ID field name (see below). Example: `LDAP-PERSON-BASE-BINDDN="cn=users,dc=mydomain,dc=com"`					
LDAP-REPOSITORY-TYPE	OpenLDAP | ActiveDirectory | SunOneDirectory | Tivoli | Novell | ApacheDS	O		u	w	
	Use predefined known fields for the respective repository type. Specify the repository type that most closely matches your actual repository. In the case of Windows Active Directory, the user ID is typically in the form *domainName**userId*.					
LDAP-SASL-AUTHENTICATION	NO | YES	O			w	
	Specifies whether or not Simple Authentication and Security Layer (SASL) is to perform the authentication check. In practice, this determines whether or not the password supplied by the user is passed in plain text between the broker kernel and the LDAP server. If SASL is activated, this implies that the password is encrypted. NO Password is sent to LDAP server in plain text. YES Password is sent to LDAP server encrypted.					
LDAP-USERID-FIELD	cn | *uidFieldName*	O		u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	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			
	Determines whether the password and new password are converted to uppercase before verification.					
PRODUCT	RACF | ACF2 | TOP-SECRET	O	z			
	Specifies the name of the installed security product. This attribute is used to analyze security-system-specific errors. The following systems are currently supported: ACF2 Security system ACF2 is installed. RACF Security system RACF is installed. Default. TOP-SECRET Security system TOP-SECRET is installed. The default value is used if an incorrect or no value is specified.					
PROPAGATE-TRUSTED-USERID	YES | NO	O	z			
	Determines whether a client user ID obtained by means of the trusted user ID mechanism is propagated to a server using the ACI field CLIENT-USERID.					
SAF-CLASS	NBKSAG | *SAFClassName*	O	z			
	Specifies the name of the SAF class/type used to hold the EntireX-related resource profiles.					
SAF-CLASS-IP	NBKSAG | *SAFClassName*	O	z			
	Specifies the name of the SAF class/type used when performing IP address authorization checks.					
SECURITY-LEVEL	AUTHORIZATION | AUTHENTICATION	O	z	u	w	b
	Specifies the mode of operation. AUTHORIZATION Authorization and authentication (not under BS2000). AUTHENTICATION Authentication. **Note:** In version 8.0, the default value for this parameter was AUTHORIZATION.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
SECURITY-NODE	YES | *name*	O	z			
	This parameter can be used to specify a prefix that is added to all authorization checks, enabling different broker kernels, in different environments, to perform separate authorization checks according to each broker kernel. For example, it is often important to distinguish between production, test, and development environments. YES This causes the broker ID to be used as a prefix for all authorization checks. *name* This causes the actual text (maximum 8 characters) to be prefixed onto all authorization checks. **Note:** By *not* setting this parameter, no prefix is added to the resource check (the default behavior).					
SECURITY-SYSTEM	OS | LDAP	O	z	u	w	b
	OS Authentication is performed against the local operating system. Default if SECURITY=YES is specified and section DEFAULTS=SECURITY is omitted from the attribute file. LDAP Authentication and authorization are performed against the LDAP repository specified under LDAP-AUTHENTICATION-URL and LDAP-AUTHORIZATION-URL.					
TRACE-LEVEL	0-4	O	z	u	w	b
	Trace level for EntireX Security. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log security violations and access denied/permitted. 2 All of trace level 1, plus internal errors. 3 All of trace level 2, plus function entered/exit messages with argument values and some progress messages. 4 All of trace level 3, plus some selected data areas for problem analysis. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD. **Note:** Setting this value also affects tracing for authorization rules.					
TRUSTED-USERID	YES | NO	O	z			
	Activates the trusted user ID mechanism for broker requests arriving over the local Adabas IPC mechanism.					
USERID-TO-	NO | YES	O	z			

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
UPPER-CASE	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	BS2000
CERT-AUTHENTICATION	NO | YES	O	z			
	NO Do not use SSL certificates for authentication. YES Use corresponding port for certificate-based authentication. See *Using SSL Certificates for Authentication* in the EntireX Security documentation for z/OS.					
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O	z	u	w	b
	Non-activity of the TCP/IP connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity test is disabled. On the stub side, non-activity can be set with the environment variable ETB_NONACT. See *Limiting the TCP/IP Connection Lifetime* under z/OS | UNIX | Windows | z/VSE in the platform-specific *Administering Broker Stubs* documentation.					
HOST	0.0.0.0 | *hostname* | *IP address*	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 broker's TCP/IP transport communicator.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	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	b

	The TCP/IP port number on which the broker will listen for connection requests.
	
	If not specified, the broker will attempt to find its TCP/IP port number from the TCP/IP services file, using `getservbyname`. If it cannot find the number here, the default value of 1971 is used.
	
	A maximum of five `HOST/PORT` pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.
	
	Example for multiple ports on z/OS:
	
	`HOST=localhost,PORT=3930`
	`HOST=0.0.0.0,PORT=3931`
	
	■ Port 3930 is used for *local* TCP/IP communication only and is not visible outside the z/OS host.
	■ Port 3931 is used for *global* TCP/IP communication. With IBM's AT-TLS this port is turned into a TLS port, see *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
	
	With this configuration you can reach the broker from outside the z/OS host via the secure TLS connection only (port 3931). The TCP connection (port 3930) can only be used from inside the z/OS host.					

RESTART	YES | NO	O	z	u	w	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. This setting applies to all TCP/IP communicators.					
RETRY-LIMIT	20 | *n* | UNLIM	O	z	u	w	b
	Maximum number of attempts to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.					
RETRY-TIME	3M | *n* | *n*S | *n*M | *n*H	O	z	u	w	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).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	*n*H Wait time in hours (max. 596523). Minimum wait time is 1S. This setting applies to all TCP/IP communicators.					
REUSE-ADDRESS	YES | NO	O	z	u		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. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

c-tree-specific Attributes

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

Not available under z/OS or BS2000.

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

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

SSL/TLS-specific Attributes

The Broker can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term “SSL” in this section refers to both SSL and TLS. RPC-based clients and servers, as well as ACI clients and servers, are always SSL clients. The broker is always the SSL server. For an introduction see *SSL/TLS, HTTP(S), and Certificates with EntireX*. Your operating system determines whether this section of the attribute file is required:

- **z/OS**
  The SSL-specific attribute section is not used. You can use IBM's Application Transparent Transport Layer Security (AT-TLS).
  See *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
- **UNIX and Windows**
  The SSL-specific attribute section is required, and begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.
  The attributes in this section are needed to execute the SSL communicator of the EntireX Broker kernel.
  See also *Running Broker with SSL/TLS Transport* under UNIX | Windows.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
`CIPHER-SUITE`	*`string`*	O		u	w	b
	String that is passed to the underlying SSL/TLS implementation. SSL/TLS is a standardized protocol that uses different cryptographic functions (hash functions, symmetric and asymmetric encryption etc.). Some of these must be implemented in the SSL/TLS stack; others are optional. When an SSL/TLS connection is created, both parties agree by "handshake" on the cipher suite, that is, the algorithms and key lengths used. In a default scenario, this information depends on what both sides are capable of. It can be influenced by setting the attribute `CIPHER-SUITE` for the SSL/TLS server side (the broker always implements the server side). Thus stubs connect to the broker and thereby become the SSL/TLS clients. Under UNIX, Windows and BS2000, the OpenSSL implementation is used. The SSL protocol is obsolete. It is no longer available. The TLS protocol is the successor of SSL and is readily available in OpenSSL. The default OpenSSL configuration uses FIPS 140-2 approved cipher suites, eligible for TLS v1.2, but without anonymous Diffie-Hellman (ADH) and pre-shared key (PSK) algorithms. The resulting set of cipher suites provides for authentication and strong encryption: `CIPHER-SUITE=FIPS+TLSv1.2:!ADH:!PSK:@STRENGTH`					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	See *https://www.openssl.org/docs/man1.1.1/man1/ciphers*.					
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O		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		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-FILE	*filename*	R		u	w	b
	File that contains the broker's private key (if not contained in KEY-STORE). For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: MyAppKey.pem. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).					
KEY-PASSWD	*password* (A32)	R		u	w	b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. Deprecated. See KEY-PASSWD-ENCRYTPED below.					
KEY-PASSWD-ENCRYPTED	*encrypted value* (A64)	R		u	w	b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. This attribute replaces KEY-PASSWD to avoid a clear-text password as attribute value. If KEY-PASSWD and KEY-PASSWD-ENCRYTPED are both supplied, KEY-PASSWD-ENCRYTPED takes precedence. Use program etbnattr to get the encrypted password:					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	`etbnattr -w ssl_key_password --echo_password_only` This writes the encrypted password to standard output.					
KEY-STORE	*filename*	R		u	w	b
	SSL certificate; may contain the private key. For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: *ExxAppCert.pem*. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O		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		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 sample attribute file. If the port number is not specified, the broker will use the default value of 1958.					
RESTART	YES | NO	O		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		u	w	b
	Maximum number of attempts to restart the SSL communicator.					
RETRY-TIME	3M | *n* | *n*S | *n*M | *n*H	O		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		u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	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		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		u	w	b
	The level of tracing to be performed while the broker is running with transport method SSL/TLS. It overrides the global value of trace level for all SSL/TLS routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					
TRUST-STORE	*filename*|*keyring*	R		u	w	b
	Location of the store containing certificates of trust Certificate Authorities (or CAs). Specify the file name of the CA certificate store. Examples: EXXCACERT.PEM, C:\Certs\ExxCACert.pem					
VERIFY-CLIENT	NO | YES	O		u	w	b
	YES Additional client certificate required. NO No client certificate required (default).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
	For more information see *SSL/TLS, HTTP(S), and Certificates with EntireX*.					

DIV-specific Attributes

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

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

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

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

Adabas-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
`BLKSIZE`	`126-20000`	O	z	u	w	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	b
	Database ID of Adabas database where the persistent store resides.					
`FNR`	`1-32535`	R	z	u	w	b
	File number of broker persistent store file.					
`FORCE-COLD`	`N` | `Y`	O	z	u	w	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	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	b
	Determines whether driver for Adabas persistent store is to issue an `OPEN` command to Adabas.					
`SVC`	`200-255`	R	z			
	Use this parameter to specify the Adabas SVC number to be used by the Adabas persistent store driver.					

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

Application Monitoring-specific Attributes

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

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

Authorization Rule-specific Attributes

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	UNIX	Windows	BS2000
`RULE-NAME`	`A32`	R		u	w	
	Specifies a rule name. A rule is a container for a list of services and a list of client and server user IDs. All users defined in a rule are authorized to use all services defined in this rule. See example under *Rules Stored in Broker Attribute File*.					

`CLASS`
`SERVER`
`SERVICE`
	`A32`	R		u	w	

	These three attributes together identify the service. `CLASS` must be specified first, followed immediately by `SERVER` and `SERVICE`. ***Wildcard Service Definitions*** are allowed.					
`CLIENT-USER-ID`	`A32`	R		u	w	
	Defines an authorized client user ID.					
`SERVER-USER-ID`	`A32`	R		u	w	
	Defines an authorized server user ID.					

Variable Definition File

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

The following attributes are considered unique for each machine:

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

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

4 Configuring Broker for Internationalization

Software internationalization is the process of designing products and services so that they can be adapted easily to a variety of different local languages and cultures. Internationalization within EntireX means internationalization of messages: the incoming and outgoing messages are converted to the desired codepage of the platform in use. This chapter explains in detail how to configure the broker for character conversion.

See also *Internationalization with EntireX*.

Configuring ICU Conversion

To configure ICU conversion

1 In the Broker attribute file, set the service-specific attribute `CONVERSION`. Examples:

- ICU Conversion with SAGTCHA for *ACI-based Programming*:

```
CONVERSION=(SAGTCHA,OPTION=SUBSTITUTE)
```

- ICU Conversion with SAGTRPC for *RPC-based Components* and *Reliable RPC*:

```
CONVERSION=(SAGTRPC,OPTION=STOP)
```

2 Optionally configure a `CONVERSION OPTION` to tune error behavior to meet your requirements; see *`OPTION` Values for Conversion*.

3 For the Broker attribute, check if ICU conversion is possible, that is, the attribute `ICU-CONVERSION` is either

- not defined, its default is `YES`
- set to `YES`

To configure locale string defaults (optional)

- If the broker's locale string defaults do not match your requirements (see *Broker's Locale String Defaults*), we recommend you assign suitable locale string defaults for your country and region, see the respective attribute in *Codepage-specific Attributes* for how to customize the broker's locale string defaults.

To customize mapping of locale strings (optional)

- If the built-in locale string mapping mechanism does not match your requirements, you can assign specific codepages to locale strings. See *Broker's Built-in Locale String Mapping* and `locale-string` for information on customizing the mapping of locale strings to codepages.

Building and Installing ICU Custom Converters

User-written ICU custom-converters can be used for *ACI-based Programming*, *RPC-based Components*, and *Reliable RPC*. This section covers the following topics:

Writing a User-written ICU Converter

ICU uses algorithmic conversion, non-algorithmic conversion and combinations of both. See *ICU Conversion*. Non-algorithmic converters defined by the UCM format are the easiest way to define user-written ICU converters. See *UCM Format*.

To write a (non-algorithmic) user-written ICU converter

- Define the ICU converter file in UCM format using a text editor to meet your requirements.

  Note: For further explanation of the UCM file format, see *ICU Resources*.

Writing algorithmic and partially algorithmic converters can be complex. However, they can be installed into EntireX in the same way as the table-driven, non-algorithmic ones. A description of how to write algorithmic and partially algorithmic converters is beyond the scope of this documentation. See the ICU documentation and other sources specified under *ICU Resources* for more information.

Compiling a User-written ICU Converter

To compile the user-written ICU converter

1. Extract the ICU tool `makeconv` and ICU shared libraries as described under *Installing the EntireX ICU Custom Converter Build Environment under z/OS UNIX*.
2. Compile the converter source files (extension *.ucm*) into binary converter files (extension ".cnv") using the ICU tool `makeconv`. Example:

```
makeconv -v myebcdic.ucm
```

This produces a binary converter file named *myebcdic.cnv*.

Caution: The binary format "cnv" depends on the endianness (big/little-endian) and character set family (ASCII/EBCDIC) of the computer where it is produced. For example, a binary converter file produced on a machine with big endianes cannot be executed on a machine with little-endian (and vice versa) or character set family *EBCDIC* cannot be executed on a machine with character set family *ASCII* (and vice versa). It is highly recommended to compile the converter source file(s) on the same target platform where the broker runs - otherwise unpredictable result may occur.

Installing a User-written ICU Converter

To install the user-written ICU converter

1 Define the broker attribute `ICU-DATA-DIRECTORY`. See *Broker-specific Attributes*.

Example:

```
ICU DATA DIRECTORY="/home/sag/EntireX/config/etb"
```

2 Define the subdirectory *icudt<icu-version><endianness>* within the `ICU-DATA-DIRECTORY`

where <*icu-version*> is the ICU version used, for example 54, and
<*endianness*> is “e” EBCDIC (big-endian)

Example:

```
/home/sag/EntireX/config/etb/icudt54e
```

Notes:

1. The subdirectory and its naming are given by ICU standard. It is not invented by Software AG.
2. See the Release Notes to determine the ICU version used by the broker you are running and form the correct name - otherwise the user-written ICU converter will not be located.
3. There are also other approaches supported by ICU to locate converters. These approaches are (also) ICU version dependent. However, Software AG recommends the mechanism described above. See the ICU website for more information under *ICU Resources*.

3 Copy the user-written ICU converter binary file (extension "cnv") to the referenced by `ICU-DATA-DIRECTORY` and its subdirectory defined under steps 1 and 2 above. Examples:

```
/home/sag/EntireX/config/etb/icudt54e/myebcdic.cnv
/home/sag/EntireX/config/etb/icudt54e/myascii.cnv
```

4 If the converter name is not sent as the locale string by your application, customize the mapping of locale strings by assigning the user-written ICU converter (codepage) to locale strings in the Broker attribute file, see `locale-string` for how to customize the mapping of locale strings to codepages. Example:

```
DEFAULTS=CODEPAGE
/* Customer-written ICU converter */
CP1140=myebcdic
CP0819=myascii
```

5 For the Broker attribute, check whether ICU conversion is possible, that is, the attribute `ICU-CONVERSION` is not defined (default=`YES`) or set to `YES`.

6 For the Broker attribute, check whether use of ICU custom converters is possible, that is, the attribute `ICU-SET-DATA-DIRECTORY` is not defined (default=`YES`) or set to `YES`.

Writing Translation User Exits

This section covers the following topics:

Introduction

EntireX Broker provides an interface to enable user-written translation routines in the programming language Assembler. It contains three parameters:

- The address of the TRAP control block (TRAP = Translation Routine / Area for Parameters).
- The address of a temporary work area. It is aligned to fullword / long integer boundary (divisible by 4). The work area can only be used for temporary needs and is cleared after return.
- A fullword (long integer) that contains the length of the work area.

Note: Names for user-written translation routines starting with "SAG" are reserved for Software AG usage and must not be used, e.g. "SAGTCHA" and "SAGTRPC".

Structure of the TRAP Control Block

The Assembler dummy section TR$TRAP covers the layout of the TRAP control block:

```
TR$TRAP  DSECT ,
TR$TYPE  DS       F            TRAP type
TR$TYP2  EQU      2            TRAP type ETB 121
TR$ILEN  DS       F            Input buffer length
TR$IBUF  DS       A            Address of input buffer
TR$OLEN  DS       F            Output buffer length
TR$OBUF  DS       A            Address of output buffer
TR$DLEN  DS       F            Length of data returned:
*                              Should be set to the minimum value of TR$ILEN
*                              and TR$OLEN.
TR$SHOST DS       F            Sender's host:
*                              x'00000000' = little endian
*                              x'00000001' = big endian
TR$SCODE DS       F            Sender's character set:
SEBCIBM  EQU      X'00000022'  EBCDIC (IBM)
SEBCSNI  EQU      X'00000042'  EBCDIC (SNI)
SA88591  EQU      X'00000080'  ASCII
TR$RHOST DS       F            Receiver's host     --> see TR$SHOST
TR$RCODE DS       F            Receiver's char set --> see TR$SCODE
TR$BHOST DS       F            BROKER host         --> see TR$SHOST
TR$BCODE DS       F            BROKER char set     --> see TR$SCODE
TR$SENVA DS       F            Sender's ENVIRONMENT field supplied:
OFF      EQU      X'00000000'  ENVIRONMENT field not set
ON       EQU      X'00000001'  ENVIRONMENT field set
*
TR$RENVA DS       F            Receiver's ENVIRONMENT field supplied:
*                              --> see TR$SENVA
TR$SENV  DS       CL32         Sender's ENVIRONMENT field
TR$RENV  DS       CL32         Receiver's ENVIRONMENT field
TR$LEN   EQU      *-TR$TRAP    Length of TRAP
```

Using the TRAP Fields

The TR$DLEN must be supplied by the user-written translation routine. It tells the Broker the length of the message of the translation. In our example its value is set to the minimum length of the input and output buffer.

All other TRAP fields are supplied by the Broker and must not be modified by the user-written translation routine.

The incoming message is located in a buffer pointed to by TR$IBUF. The length (not to be exceeded) is supplied in TR$ILEN. The character set information from the send buffer can be taken from TR$SCODE.

The outgoing message must be written to the buffer pointed to by TR$OBUF. The length of the output buffer is given in the field TR$OLEN. The character set is specified in TR$RCODE. If the addresses

given in TR$IBUF and TR$OBUF point to the same location, it is not necessary to copy the data from the input buffer to the output buffer.

The environment fields TR$SENVA and TR$RENVA are provided to handle site-dependent character set information. For the SEND and/or RECEIVE functions, you can specify data in the ENVIRONMENT field of the Broker ACI control block. This data is translated into the codepage of the platform where EntireX Broker is running (see field TR$BCODE) and is available to the TR$SENV or TR$RENV field in the TRAP control block. TR$SENVA or TR$RENVA are set to ON if environmental data is available. Any values given in the API field ENVIRONMENT must correspond to the values handled in the translation routine.

To assemble and link the SAGTCHA user-written translation routine

- Assemble and link your translation routine. You can give the resulting load module any name that does not begin with "SAG". Names starting with "SAG", such as "SAGTCHA", are reserved for Software AG.

Configuring Translation User Exits

To configure translation user exits

As a prerequisite, the user-written translation module must be accessible to the Broker worker threads.

1. Copy the user-written translation module into any library of the Broker's steplib concatenation.
2. In the Broker attribute file, set the service-specific attribute TRANSLATION to the name of the user-written translation routine. Example:

```
TRANSLATION=MYTRANS
```

Writing SAGTRPC User Exits

This section covers the following topics:

Introduction

EntireX Broker provides an interface to SAGTRPC user exit routines written in the programming language Assembler. The interface contains three parameters:

- The address of the UE (user exit) control block.
- The address of a temporary work area. It is aligned to a fullword / long-integer boundary (divisible by 4). The work area can only be used temporarily and is cleared after return.
- A fullword (long integer) that contains the length of the work area.

Note: Names for conversion routines starting with "SAG" are reserved for Software AG usage and must not be used, e.g. "SAGTCHA" and "SAGTRPC".

Structure of the User Exit Control Block

The Assembler dummy section `UE$CB` shows the layout of the user exit control block.

```
UE$CB    DSECT , ...... User Exit Control Block
*                      **********************
*                                                    Direction
*                                                    ---------
UE$VERS  DS    F       UECB version                   input
UE$VER1  EQU   1       UECB version 1
UE$IBUF  DS    A       Address of input buffer        input
UE$ILEN  DS    F       Input buffer length            input
UE$OBUF  DS    A       Address of output buffer       input
UE$OLEN  DS    F       Output buffer length           input
UE$DLEN  DS    F       Length of data returned        output
*
UE$SHOST DS    F       Senders host:                  input
*                      x'00000000' = little endian
*                      x'00000001' = big endian
*
UE$SCODE DS    F       Senders character set:         input
SEBCIBM EQU X'00000022' EBCDIC (IBM)
SEBCSNI EQU X'00000042' EBCDIC (SNI)
SA88591 EQU X'00000080' ASCII
*
UE$RHOST DS    F       Receivers host      --> see UE$SHOST  input
UE$RCODE DS    F       Receivers char set  --> see UE$SCODE  input
UE$BHOST DS    F       BROKER host         --> see UE$SHOST  input
UE$BCODE DS    F       BROKER char set     --> see UE$SCODE  input
*
UE$SCP   DS    F       Sender   Codepage number
UE$RCP   DS    F       Receiver Codepage number
UE$BCP   DS    F       Broker   Codepage number
*
UE$FCT   DS    CL1     Function                       input
FCTCONV  EQU   C'C'    Function CONVERT
```

```
FCTGLEN  EQU   C'L'     Function GETLENGTH
UE$DIR   DS    CL1      Direction                              input
DIRS2B   EQU   C'1'     Direction Sender to Broker
DIRS2R   EQU   C'2'     Direction Sender to Receiver
DIRB2R   EQU   C'3'     Direction Broker to Receiver
UE$FMT   DS    CL2      Format                                 input
FMTUSER  EQU   C'01'    User Data like User ID, Program, Library
FMTMETA  EQU   C'02'    Meta Data Header
FMTFB    EQU   C'03'    Format Buffer
FMTSB    EQU   C'04'    String Buffer
FMTVBN   EQU   C'05'    Meta data value buffer
FMTPRE   EQU   C'99'    Preview format buffer
FMTA     EQU   C'A '    Data Type A
FMTAV    EQU   C'AV'    Data Type AV
FMTB     EQU   C'B '    Data Type B
FMTBV    EQU   C'BV'    Data Type BV
FMTD     EQU   C'D '    Data Type D
FMTF4    EQU   C'F4'    Data Type F4
FMTF8    EQU   C'F8'    Data Type F8
FMTI1    EQU   C'I1'    Data Type I1
FMTI2    EQU   C'I2'    Data Type I2
FMTI4    EQU   C'I4'    Data Type I4
FMTK     EQU   C'K '    Data Type K
FMTKV    EQU   C'KV'    Data Type KV
FMTL     EQU   C'L '    Data Type L
FMTN     EQU   C'N '    Data Type N
FMTP     EQU   C'P '    Data Type P
FMTT     EQU   C'T '    Data Type T
FMTU     EQU   C'U '    Data Type U
FMTUV    EQU   C'UV'    Data Type UV
UE$ETXT  DS    CL40     Error Text output
UE$LEN   EQU   *-UE$CB  Length of UECB
         SPACE ,
```

The user-written conversion exit example `USRTRPC` is delivered in the EntireX common source library EXX107.SRCE. The related JCL to build `USRTCHA` is in member `EXBUSRXT` in the EntireX common jobs library. See *Contents of Mainframe Installation Medium*.

Using the User Exit Interface Fields

The user exit provides two separate functions, `CONVERT` and `GETLENGTH`. The field `UE$FCT` indicates the function to execute.

Errors

Both functions can send an error, using register 15 in the range 1 to 9999 to SAGTRPC together with an error text in the field UE$ETXT.

- A value of 0 returned in register 15 means successful response.
- Error 9999 is reserved for output buffer overflow. See *CONVERT Function*.
- When an error occurs, the conversion of the message will be aborted and the error text will be sent to the receiver (client or server). The error is prefixed with the error class 1011. See *Message Class 1011 - User-definable SAGTRPC Conversion Exit*.

Example:

The user exit returns 1 in register 15 and the message “Invalid Function” in UE$ETXT. The receiver gets the error message 10110001 Invalid Function.

CONVERT Function

This function has to be executed when the contents of UE$FCT match the definition FCTCONV.

UE$DLEN must be supplied by SAGTRPC's user-written conversion exit. Its value must be set to the length of the output buffer.

All other interface fields are supplied by the Broker and must not be modified by SAGTRPC's user-written conversion exit.

The incoming data is located in a buffer pointed to by UE$IBUF. UE$ILEN defines the length.

The outgoing converted message must be written to the buffer pointed to by UE$OBUF. The field TR$OLEN defines the maximum length available.

For variable length data such as AV and KV, an output buffer overflow can occur if the message size increases after conversion or the receiver's receive buffer is too small. In this case error 9999 “output buffer overflow” must be returned, which calls the *GETLENGTH Function* for the remaining fields.

GETLENGTH Function

The GETLENGTH function evaluates the needed length of the output buffer after conversion. An actual conversion must not be performed. The length needed must be returned in the field UE$OLEN.

The GETLENGTH function is called for remaining fields after the CONVERT function returned the error 9999 “output buffer overflow”.

The purpose of this function is to evaluate the length needed by the receiver's receive buffer. This length is returned to the receiver in the ACI field RETURN-LENGTH. The receiver can then use the

Broker ACI function RECEIVE with the option LAST together with a receive buffer large enough to reread the message.

Character Set and Codepage

The character-set information used is the same as in the user-written translation routine and is taken from UE$SCODE (for the sender), UE$RCODE (for the receiver) and UE$BCODE (for the Broker). The character-set information depends on the direction information given in the field UE$DIR. See the following table:

UE$DIR	From Character Set	To Character Set
DIRS2B (Sender to Broker)	UE$SCODE	UE$BCODE
DIRS2R (Sender to Receiver)	UE$SCODE	UE$RCODE
DIRB2R (Broker to Receiver)	UE$BCODE	UE$RCODE

Alternatively, the codepage as derived from the locale string mapping process is provided in UE$SCP (sender codepage), UE$RCP (receiver codepage) and UE$BCP (Broker codepage), and can be used to find the correct conversion table. See the following table and also *Locale String Mapping*.

UE$DIR	From Codepage	To Codepage
DIRS2B (Sender to Broker)	UE$SCP	UE$BCP
DIRS2R (Sender to Receiver)	UE$SCP	UE$RCP
DIRB2R (Broker to Receiver)	UE$BCP	UE$RCP

Software AG IDL Data Types to Convert

The field UE$FMT provides the SAGTRPC user-written conversion exit with the information on the IDL data types to convert. Each data type can be handled independently.

UE$FMT	Data to be converted	Notes
FMTA	IDL data type A	1, 3, 4
FMTAV	IDL data type AV	4, 5
FMTB	IDL data type B	1, 2, 7
FMTBV	IDL data type BV	1, 2, 7
FMTD	IDL data type D	1, 2, 7
FMTF4	IDL data type F4	1, 2, 7
FMTF8	IDL data type F8	1, 2, 7
FMTI1	IDL data type I1	1, 2, 7
FMTI2	IDL data type I2	1, 2, 7

UE$FMT	Data to be converted	Notes
FMTI4	IDL data type I4	1, 2, 7
FMTK	IDL data type K	1, 3, 4
FMTKV	IDL data type KV	4, 5
FMTL	IDL data type L	1, 2, 7
FMTN	IDL data type N	1, 2, 7
FMTP	IDL data type P	1, 2, 7
FMTT	IDL data type T	1, 2, 8
FMTU	IDL data type U	1, 2, 7
FMTUV	IDL data type UV	1, 2, 7
FMTUSER	RPC user data such as user ID, library, program...	1, 3, 4
FMTMETA	RPC metadata	1, 2, 7
FMTFB	RPC format buffer	1, 2, 7
FMTSB	RPC metadata variable length	4, 5, 7
FMTPRE	Preview data	4, 6, 7

Notes:

1. Field length is constant.
2. The field content length must not increase or decrease during conversion. If this happens, the user exit should produce an error.
3. If the field content length *decreases* during the conversion, suitable padding characters (normally blanks) have to be used.
   If the field content length *increases* during conversion and exceeds the field length, the contents must be truncated or, alternatively, the conversion can be aborted and an error produced.
4. If the contents are truncated, character boundaries are the responsibility of the user exit. Complete valid characters after conversion have to be guaranteed. This may be a complex task for code-pages described under *Arabic Shaping*, *EBCDIC Stateful Codepages* or *Multibyte or Double-byte Codepages*. For single-byte codepages it is simple because the character boundaries are the same as the byte boundaries.
5. The field length can decrease or increase during the conversion up to the output buffer length. The new field length must be returned in UE$DLEN. If the output buffer in the CONVERT function is too small, error 9999 must be returned to the caller.
6. The field buffer should continue to be converted until the output buffer is full or the input buffer has been processed. If the field content length increases or truncations occur, no error should be produced. If the field content length decreases, there should be no padding. The new field length should simply be returned to the caller.

7. Codepages used for RPC data streams must meet several requirements. See *Codepage Requirements for RPC Data Stream Conversions*. If these are not met, the codepage cannot be used to convert RPC data streams.

To assemble and link the SAGTRPC user-written conversion exit

- Assemble and link your conversion exit. You can give the resulting load module any name that does not begin with "SAG". Names starting with "SAG", such as "SAGTRPC", are reserved for Software AG. Refer to the JCL provided in member EXBUSRXT in data set EXX107.JOBS.

To install and configure the SAGTRPC user-written conversion exit

- Refer to the instructions under ***Configuring SAGTRPC User Exits***.

Configuring SAGTRPC User Exits

To configure SAGTRPC user exits

1 The user-written conversion module must be accessible to the Broker worker threads. Therefore, copy the user-written conversion module into any library of the Broker's steplib concatenation.

2 In the Broker attribute file, set the service-specific attribute `CONVERSION` to the name of the user-written SAGTRPC user exit routine. Example:

```
CONVERSION=(MYTRANS)
```

To configure locale string defaults

- If the broker's locale string defaults do not match your requirements, we recommend you assign suitable locale string defaults for your country and region. See the appropriate attribute under *Codepage-specific Attributes* for information on customizing broker's locale string defaults, and also *Locale String Mapping*.

To customize mapping of locale strings

- If the broker's built-in locale string mechanism does not match your requirements, you can assign specific codepages to locale strings. See *Broker's Built-in Locale String Mapping* and the appropriate attribute under *Codepage-specific Attributes* for information on customizing broker's locale string defaults.

5 Managing the Broker Persistent Store

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

Under z/OS, the broker persistent store can be implemented with:

- the Adabas database of Software AG
- a VSAM linear data set (LDS) accessed using Data In Virtual (DIV)

See also *Concepts of Persistent Messaging*.

Implementing an Adabas Database as Persistent Store

Introduction

EntireX provides an Adabas persistent driver. This enables Broker unit of work (UOW) messages and their status to be stored in an Adabas file. It is designed to work with Adabas databases under z/OS, UNIX, Windows, BS2000 and z/VSE, and can be used where the database resides on a different machine to Broker kernel. For performance reasons, we recommend using EntireX Broker on the same machine as the Adabas database.

Configuring and Operating the Adabas Persistent Store

Selecting the Adabas Persistent Store Driver

The Adabas persistent store driver module is contained within the regular Broker load library or binaries directory. Module `ADAPSI` is activated by specifying the `PSTORE-TYPE` parameter. Use the supplied job EXBJ015 from data set EXX107.JOBS to define and install the persistent store file in your Adabas database. This job creates and loads the Adabas file into the database.

Restrictions

If a HOT start is performed, the Broker kernel must be executed on the same platform on which also the previous Broker executed. This is because some portions of the persistent data are stored in the native character set and format of the Broker kernel. It is also necessary to start Broker with the same Broker ID as the previous Broker executed.

If a COLD start is executed, a check is made to ensure the Broker ID and platform information found in the persistent store file is consistent with the Broker being started (provided the persistent store file is not empty). This is done to prevent accidental deletion of data in the persistent store by a different Broker ID. If you intend to COLD start Broker and to utilize a persistent store file which has been used previously by a different Broker ID, you must supply the additional `PSTORE-TYPE` parameter `FORCE-COLD=Y`.

Recommendations

- Perform regular backup operations on your Adabas database. The persistent store driver writes C1 checkpoint records at each start up and shut down of Broker.
- Significant performance improvements can be achieved using Adabas/Fastpath where available. See Adabas/Fastpath documentation for details of installation and configuration of Adabas/Fastpath.
- For performance reasons, execute Broker on the same machine as Adabas.

Broker Checkpoints in Adabas

During startup, Broker writes the following C1 checkpoint records to the Adabas database. The time, date and job name are recorded in the Adabas checkpoint log. This enables Adabas protection logs to be coordinated with Broker executions. This information can be read from Adabas, using the `ADAREP` utility with option `CPLIST`:

Broker Execution Name	Broker Execution Type	Adabas
ETBC	Broker Cold Start	Normal Cold Start
ETBH	Broker Hot Start	Normal Hot Start
ETBT	Broker Termination	Normal Termination

Adabas DBA Considerations

BLKSIZE : Adabas Persistent Store Parameter for Broker

Caution should be exercised when defining the block size (`BLKSIZE`) parameter for the Adabas persistent store. This determines how much UOW message data can be stored within a single Adabas record. Therefore, do not define a much larger block size than the size of the maximum unit of work being processed by Broker. (Remember to add 41 bytes for each message in the unit of work.) The advantage of having a good fit between the unit of work and the block size is that fewer records are required for each I/O operation.

It is necessary to consider the following Adabas parameters and settings when using Adabas for the persistent store file:

Table of Adabas Parameter Settings

Topic	Description
Allowing Sufficient Adabas UQ Elements	Allow sufficient Adabas user queue (UQ) elements each time you start Broker. The Broker utilizes a number of user queue elements equal to the number of worker tasks (`NUM-WORKER`), plus two. Adabas timeout parameter (`TNAE`) determines how long the user queue elements will remain. This can be important if Broker is restarted after an abnormal termination, and provision must be made for sufficient user queue elements in the event of restarting Broker. Sample JCL SAGJ014 is provided in data set EXX107.JOBS for z/OS to enforce clean-up of any user queue element belonging to the previous Broker job. This JCL can be inserted into the job step before starting up Broker.
Setting Size of Hold Queue Parameters	Consideration must be given to the Adabas hold queue parameters `NISNHQ` and `NH`. These must be sufficiently large to allow Adabas to add/update/delete the actual number of records within a single unit of work. Example: where there are 100 message within a unit of work and the average message size is 10,000 bytes, the total unit of work size is 1 MB. If, for example, a 2 KB block size (default `BLKSIZE=2000`) is utilized by the Adabas persistent store driver, there will be 500 distinct records within a single Adabas commit (`ET`) operation, and provision must be made for this to occur successfully.

Topic	Description
Setting Adabas TT Parameter	Consideration must be given to the Adabas transaction time (`TT`) parameter for cases where a large number of records is being updated within a single unit of work.
Defining LWP Size	Sufficient logical work pool (`LWP`) size must be defined so that the Adabas persistent store can update and commit the units of work. Adabas must be able to accommodate this in addition to any other processing for which it is used.
Executing Broker Kernel and Adabas Nucleus on Separate Machines	If Broker kernel is executed on a separate machine to the Adabas nucleus, with a different architecture and codepage, then we recommend running the Adabas nucleus with the `UEC` (universal conversion) option in order to ensure that Adabas C1 checkpoints are legible within the Adabas checkpoint log.
Setting `INDEXCOMPRESSION=YES`	This Adabas option can be applied to the Adabas file to reduce by approximately 50% the amount of space consumed in the indexes. This is the default setting in job EXBJ015, which is supplied in data set EXX107.JOBS to define the Adabas persistent store file.
4-byte ISNs	If you anticipate having more than 16 million records within the persistent store file, you must use 4-byte ISNs when defining the Adabas file for EntireX.
Specification of Adabas LP Parameter	**Caution:** This parameter must be specified large enough to allow the largest UOW to be stored in Adabas. If this is not large enough, Broker will detect an error (response 9; subresponse - 4 bytes - X'0003',C'LP') and Broker will not be able to write any further UOWs. See the description of the `LP` parameter under *ADARUN Parameters* in the*DBA Reference Summary* of the Adabas documentation.

Estimating the Number of Records to be Stored

To calculate the Adabas file size it is necessary to estimate the number of records being stored. As an approximate guide, there will be one Adabas record (500 bytes) for each unprocessed unit of work, plus also *n* records containing the actual message data, which depends on the logical block size and the size of the unit of work. In addition, there will be one single record (500 bytes) for each unit of work having a persisted status.

Always allow ample space for the Adabas persistent store file since the continuous operation of Broker relies of the availability of this file to store and retrieve information.

Note: If the Adabas file space is exceeded, no new units of work will be accepted.

Estimating the Number of Records to be Stored

In this example there are 100,000 Active UOW records at any one time. Each of these is associated with two message records containing the message data. UOW records are 500 bytes in length. Each message record contains 2,000 bytes. In addition, there are 500,000 UOW status records residing in the persistent store, for which the UOW has already been completely processed. These are 500 bytes long.

Note: The actual size of the data stored within the UOW message records is the sum of all the messages within the UOW, plus a 41-byte header for each message. Therefore, if the average message length is 59 bytes, the two 2,000 bytes, messages records, could contain n = 4,000 / (59+41), or 40 messages. Adabas is assumed to compress the message data by 50% in the example (this can vary according to the nature of the message data).

3-byte ISNs and RABNs are assumed in this example. A device type of 8393 is used; therefore, the ASSO block size is 4,096, and DATA block size is 27,644. Padding factor of 10% is specified.

The following example calculates the space needed for Normal Index (NI), Upper Index (UI), Address Converter (AC) and Data Storage (DS).

Calculation Factors	Required Space
■ Number entries for descriptor WK (21-byte unique key)	■ = number UOW records: 0.1 + 0.5 million + number message records: 0.2 million
■ NI Space for descriptor WK ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 800,000 * (3 + 21 + 2) ■ = 20,800,000 bytes ■ = 5,648 blocks
■ UI Space for descriptor WK ■ (3-byte ISN + 3-byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 5,648 * (21 + 3 + 3 + 1) ■ = 158,140 bytes ■ = 43 blocks
■ Number entries for descriptor WI (8-byte unique key)	■ = number processed UOW records: 0.5 million
■ NI Space for descriptor WI ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 500,000 * (3 + 8 + 2) ■ = 6,500,000 bytes ■ = 1,765 blocks
■ UI Space for descriptor WI ■ (3-byte ISN and 3 byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 17,649 * (8 + 3 + 3 + 1) ■ = 26,475 bytes ■ = 8 blocks

Calculation Factors	Required Space
■ Number entries for descriptor WL (96 byte key)	■ = number UOW records 0.1 + 0.5 million
■ NI Space for descriptor WL ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 600,000 * (3 + 96 + 2) ■ = 60,600,000 bytes ■ = 16,455 blocks
■ UI Space for descriptor WL ■ (3-byte ISN and 3 byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 164,548 * (96 + 3 + 3 + 1) ■ = 16,948,517 bytes ■ = 461 blocks
■ Address Converter space ■ (4,092 ASSO block)	■ = (800,000 + 1) * 3 / 4092 ■ = 587 blocks
■ Data storage for message data (2,000-byte records compressed by 50%)	■ = 0.2 million * 2000 * 0.5 = 200,000,000 bytes
■ Data storage for UOW data (2,000-byte records compressed by 50%)	■ = 0.6 million * 500 * 0.5 = 150,000,000 byte
■ Combined space required for data (27,644 DATA block 10% padding)	■ = 14,068 blocks
Entity Requiring Space	**Total Required Space**
Normal Index (NI)	= 23,868 blocks
Upper Index (UI)	= 512 blocks
Data Storage (DS)	= 14,068 blocks
Address Converter (AC)	= 587 blocks

Tips on Transports, Platforms and Versions

- **Entire Net-Work**
  If you intend to use Adabas persistent store through Entire Net-Work, see the Entire Net-Work documentation for installation and configuration details.
- **Adabas Versions**
  Adabas persistent store can be used on all Adabas versions currently released and supported by Software AG.

- **Prerequisite Versions of Entire Net-Work with Adabas**
  See the Adabas and Entire Net-Work documentation to determine prerequisite versions of Entire Net-Work to use with Adabas at your site.

Implementing a DIV Persistent Store

Note: From EntireX version 9.7, broker attribute `PSTORE-VERSION` must be set to 4 for a persistent store of type DIV. If you were using a lower version, you will need to perform a cold start. See `PSTORE=COLD` under *Broker-specific Broker Attributes*.

This section covers the following topics:

Introduction

The persistent store is implemented as a VSAM linear data set (LDS) accessed using Data In Virtual (DIV). This DIV persistent store is a container for units of work.

DIV is an access method that utilizes the system paging facilities for fast I/O to and from an LDS. Performance is best if the LDS is placed on the fastest storage device available such as those used for paging. An LDS may span multiple volumes.

The DIV persistent store has an internal structure that is formatted by EntireX Broker during a COLD start (see broker attribute `PSTORE`). This format is controlled by format parameter statements that the Broker reads from the attribute file. See *DIV-specific Attributes*.

Persistent store data sets are maintained using the IBM z/OS utility IDCAMS. See *Operations using `IDCAMS`* for more information.

Format Parameters

Note: This method of specifying persistent store format parameters is deprecated, but still supported for compatibility reasons. We recommend you use the attributes introduced with EntireX version 9.12 instead. See *DIV-specific Attributes*.

The persistent store format parameters define how a persistent store is formatted during a cold start operation and how it is accessed during all operations. These parameters are supplied in the attribute file section `DEFAULTS=DIV`. Knowledge of the application usage of units of work (UOWs) will be very helpful in selecting appropriate values for the parameters used to define the persistent store.

The persistent store format parameters file must begin with the word `DEFINE`, followed by "parameter name parameter value" specifications. Each parameter name must be separated from the parameter value by whitespace (blanks, tabs, or new lines). Comments may be added to the file.

A comment begins with /* and ends with */, just as in the C language. The parameters must be entered in uppercase. In the following parameter descriptions, lowercase is used to denote variables:

```
DEFINE STORE name
    DDNAME ddname
    DATASPACE NAME name-of-dataspace
    DATASPACE PAGES count-of-pages
```

Parameter	Value	Opt/ Req	Description
STORE	A8	R	Defines an internal name that is used to identify the persistent store.
DDNAME	A8	R	Defines the JCL DDNAME that will be used to access the persistent store.
DATASPACE NAME	A8	O	Defines the name of the data space that will be used to map the persistent store. Default value=DSPSTORE.
DATASPACE PAGES	126-524284	O	Defines the size of the dataspace used to map the persistent store (size=DATASPACE-PAGES * 4 KB). We recommend using the maximum value. Default value is 2048.

Example

```
DEFAULTS = DIV
DIV = "DEFINE STORE PSD01 DDNAME STORE01"
```

Operations using IDCAMS

This section covers the following topics:

- Defining a Persistent Store Linear Data Set
- Printing a Persistent Store for Diagnosis
- Copying a Persistent Store for Backup or Diagnosis
- Sample IDCAMS JCL

Defining a Persistent Store Linear Data Set

The following IDCAMS statement can be used to allocate the persistent store. It assumes that the local environment is using z/OS SMS for data management. SMS allows for simple definition, but it may not be used at your site or it may not provide optimal performance. You may therefore need to modify the following statement under the direction of your local system administrator:

```
DEFINE CLUSTER (NAME(dsn_pstore) -
MEGABYTES(15,5) -
SHAREOPTIONS(1,3) - /*this is required*/
LINEAR) /*this is required*/
```

where *dsn_pstore* is the DSNAME you chose for the PSTORE linear data set (LDS).

Note: The size of the linear data set (LDS) should be 16K times the value specified for `DATASPACE PAGES`. For example, 16K * 2048 = 32768K = 32M would be the LDS size needed to contain the default number of pages. Less than the required amount will cause initialization to fail; more will be unused space.

Printing a Persistent Store for Diagnosis

The following statement lists the catalog information for the linear data set:

```
LISTCAT ENTRIES(dsn_pstore) ALL
```

where *dsn_pstore* is the DSNAME you chose for the PSTORE linear data set (LDS).

The following statement prints the contents of a persistent store in dump format:

```
PRINT IDS(dsn_pstore) DUMP
```

Copying a Persistent Store for Backup or Diagnosis

```
REPRO IDS(dsn_pstore) ODS(your_backup_name)
```

Sample IDCAMS JCL

Sample JCL is provided as member IDCAMS in the installation source library. Each operation is contained in a separate DD. To select an operation, modify the `SYSIN DD DDNAME=` to the name of the DD enclosing the statements to be selected.

Migrating the Persistent Store

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

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

Note: RUN-MODE options PSTORE-LOAD and PSTORE-UNLOAD are deprecated and will not be supported in the next version of EntireX.

The migration procedure requires two Broker instances started with the RUN-MODE parameter. 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 for 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 persistence 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.

6 Broker Resource Allocation

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

General Considerations

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

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

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

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

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

Specifying Global Resources

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

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

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

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

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

Example

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

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

Restricting the Resources of Particular Services

You can restrict resource allocation for particular services in advance:

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

Example

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Specifying Attributes for Privileged Services

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

For example:

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

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

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

Maximum Units of Work

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

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

Calculating Resources Automatically

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

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

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

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

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

Number of Conversations

To calculate the total number of conversations automatically, ensure that the CONV-DEFAULT Broker attribute and the CONV-LIMIT service-specific attribute are not set to UNLIM anywhere in the attribute

file. Specify NUM-CONVERSATION=AUTO and an appropriate value for the CONV-DEFAULT Broker attribute. The total number of conversations will be calculated using the value specified for each service.

For example:

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

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

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

Number of Servers

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

For example:

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

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

Long Message Buffers

To calculate the number of long message buffers automatically, ensure that the LONG-BUFFER-DEFAULT Broker attribute and the LONG-BUFFER-LIMIT service-specific attribute are not set to UNLIM

anywhere in the attribute file. Specify NUM-LONG-BUFFER=AUTO and an appropriate value for the LONG-BUFFER-DEFAULT Broker attribute. The total number of long message buffers will be calculated using the value specified for each service.

For example:

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

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

Short Message Buffers

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

For example:

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

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

Dynamic Memory Management

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

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

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

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

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

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

Dynamic Worker Management

Dynamic worker management is a feature to handle the fluctuating broker workload without 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

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

Sample Storage Report

The following is an excerpt from a sample STORAGE report.

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

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

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

Maximum TCP/IP Connections per Communicator

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

- Note for z/OS

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

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

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

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

Note for z/OS

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

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

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

7 Administering Broker Stubs

Available Stubs

This table lists all broker stubs available under z/OS are to be used with the programming languages Assembler | C | COBOL | Natural | PL/I.

Your selection of a specific broker stub depends on the following:

- the environment (TP monitor, TSO, Batch, Natural)
- the transport method (NET or TCP)
- the availability of administration features such as trace
- the Software AG product using the stub

Environment	Supported Transport			Stub Module	Default Transport		Trace	
	NET	TCP	SSL				TCP	NET
Batch, TSO	Yes	Yes	(1)	BROKER (4)	NET	*Setting the Default Transport Method*	*Setting Trace*	
IMS (BMP)	Yes	Yes	(1)	BROKER (4)	NET			
IMS (MPP)	Yes	Yes	(1)	MPPETB (4)	NET	n/a		
CICS	Yes	Yes	(1)	CICSETB (4)	NET	n/a		
Com-plete	Yes	Yes	(1)	COMETB (4)	NET	n/a		
Natural	Yes	Yes	(1)	NATETB23 (4)	NET	n/a	*Setting Trace*	n/a(3)
Natural RPC Server	Yes	Yes	(1)	*NATETBZ* (2)	NET	n/a	n/a(3)	n/a(3)
Event Replicator for Adabas	Yes	Yes	(1)	ARFETB (2)	NET	n/a	n/a	n/a
IDMS/DC	No	Yes	(1)	IDMSETB (4)	TCP	n/a	n/a	n/a

Notes:

1. Use IBM's Application Transparent Transport Layer Security (AT-TLS). See *Using SSL/TLS with EntireX Components*.
2. These stubs may run in SRB mode and are zIIP-eligible. Refer to the related SAG product documentation for further details. Stubs capable of running in SRB mode do not support tracing.
3. For error detection, see *Tracing Broker Communication in Natural*.
4. The listed stub modules are the front ends for your application. Depending on transport and usage of *EntireX Security*, additional modules from EXX107.LOAD are loaded dynamically.

ARFETB

This reentrant stub is for exclusive use by Adabas Replication Services.

Prerequisites and Installation Notes

- No linkage is required.
- It may run in SRB mode and is zIIP-eligible.
- The load library containing ARFETB must be added to the STEPLIB chain of the ARF started task.

BROKER

BROKER is the recommended stub for any batch environment except Natural. For Natural, use `NATETB23` or *`NATETBZ`*.

Prerequisites and Installation Notes

- This stub can be used in a multithreading (subtasking) environment, provided ADAUSER is not linked to the application.
- It is recommended to load BROKER dynamically within the application program and not to link BROKER with any Adabas link routine. BROKER will attempt to load ADALNKR dynamically.
- However, if BROKER has to be statically linked, see the subsection ***Linkage*** below.
- At runtime you *must* ensure that library EXX107.LOAD is in the steplib of the application and that Adabas library WAL842.LOAD (or above) is in the steplib when using NET transport. See *Contents of Mainframe Installation Medium* in the z/OS Installation documentation.

Linkage

Choose the method most appropriate for your application:

- Method 1: Reentrant (Thread-safe)
- Method 2: Non-reentrant

Method 1: Reentrant (Thread-safe)

Link your application to module BROKER from the EntireX load library (EXX107.LOAD).

Linkage statements:

```
INCLUDE userlib(mainpgm) Main Program
INCLUDE exxlib(BROKER)  Broker stub
ENTRY mainpgm
NAME ...
```

The SVC number may be specified as part of the Broker ID, for example:

```
ETB220:SVC237:NET
```

Method 2: Non-reentrant

Link your application to module BROKER from the EntireX load library (EXX107.LOAD) and the module ADAUSER from the Adabas load library.

Linkage statements:

```
INCLUDE userlib(mainpgm) Main Program
INCLUDE exxlib(BROKER)   Broker stub
INCLUDE wallib(ADAUSER)  Adabas batch/TSO front end
ENTRY mainpgm
NAME ...
```

By linking with ADAUSER you can specify the required SVC in the Adabas DDCARD parameter of the application job, as shown below:

```
//JO2OS1   EXEC PGM=PROGRAM
//STEPLIB  DD DISP=SHR,DSN=EXX107.LOAD
//         DD DISP=SHR,DSN=WAL842.LOAD
//ETBPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DDCARD   DD *
ADARUN MODE=MULTI,PROGRAM=USER,SVC=237
/*
```

When linking the stub for use in IMS (BMP), substitute the appropriate Adabas link module for this environment.

Make sure the Adabas link routine (ADALNK/ADALNKR) does not contain any exits that assume the length of the Adabas data area UB is extended beyond its default value. Contact Software AG Support if you are unsure about this.

CICSETB

CICSETB is the recommended stub for any CICS environment except Natural. For Natural, use `NATETB23` or *`NATETBZ`*.

Prerequisites and Installation Notes

- EntireX RPC Server for z/OS CICS® uses CICSETB as default.
- It is recommended to load CICSETB dynamically within the application program.
- CICSETB will attempt to load ADACICS dynamically, using `EXEC CICS LINK PROGRAM`.
- CICSETB, CICSETB2 and EXAGLUE must be available in the CICS RPL search chain if called dynamically. Alter the CICS procedure or job, adding the EXX load library to both the STEPLIB and DFHRPL library concatenations. See also the job EXXCICS in the EXX jobs library for altering the CICS-related entries.
- CICSETB can be used with or without a CICS TWA (transaction work area).
  - **With TWA**
    At least 24 bytes of transaction work area must be defined in your CICS transaction if you choose to specify a TWA.
  - **Without TWA**
    Prerequisite is building an Adabas CICS interface, specifying `PARMTYP=ALL` for either the ADAGSET or LGBLSET MACRO (depending upon the Adabas version). There are no application changes required for using CICSETB without TWA.
  - If the name of your Adabas CICS link routine is different from "ADACICS", please use zap EXX107.ZAPS(EXX0007) to set the site-specific name of your Adabas CICS link routine. By default, the zap will change the name to "ADABAS".

Linkage

Link your application to member CICSETB from the EntireX load library (EXX107.LOAD).

Linkage statements for COBOL applications:

```
INCLUDE cicslib(DFHECI)   CICS Prolog Module
INCLUDE userlib(mainpgm)  Main Program
INCLUDE cicslib(DFHELII)  CICS Module
INCLUDE exxlib(CICSETB)   Broker stub
ENTRY mainpgm
NAME ...
```

Linkage statements for Assembler applications:

```
INCLUDE cicslib(DFHEAI)  CICS Prolog Module
INCLUDE userlib(mainpgm) Main Program
INCLUDE cicslib(DFHEAIO) CICS Module
INCLUDE exxlib(CICSETB)  Broker stub
ENTRY mainpgm
NAME ...
```

COMETB

COMETB is the stub for any Com-plete environment. For Natural, use `NATETB23` or *`NATETBZ`*.

Prerequisites and Installation Notes

- We recommend you load COMETB dynamically within the application program.
- COMETB must be available in the COMPLIB search chain if called dynamically. Modify the Com-plete procedure or job, adding the EXX load library to the COMPLIB library concatenations.
- COMETB requires about 950 KB storage above the line. Increase the Com-plete SYSPARM `THSIZEABOVE` by 950 KB.
- If the ***EXAENV Environment Store*** is used (`DD EXAENV` is defined in the Com-plete started task JCL), increase the ULIB region size for your application that calls COMETB by 4 KB storage below the line.

Linkage

It is also possible to link your application to member COMETB from the EntireX load library (EXX107.LOAD).

EXAENV Environment Store

The EXAENV Environment Store is used under Com-plete only. A partitioned data set is assigned by `DD EXAENV`. It represents the environment store for all Com-plete users. PDS members in the store are used to define variables and assign values. The PDS member name used is the name of the user logged on to Com-plete. If you want to define your own variables, add a text member with your user name and put all variables into it.

A member with the name DEFAULT may contain global variables valid for all users.

The environment store has following data set characteristics:

```
DSORG=PO
LRECL=80
RECFM=FB
```

A line in the text member setting a variable and its value looks like:

```
ETB_STUBLOG=1
```

Variable name and variable value are left-justified and delimited by an equals sign. The first blank in the line identifies the end of the environment value definition.

See also *Prerequisites and Installation Notes*.

IDMSETB

Prerequisites and Installation Notes

IDMSETB cannot be called dynamically.

Linkage

Link your application to member IDMSETB from the EntireX load library (EXX107.LOAD).

MPPETB

Prerequisites and Installation Notes

- MPPETB can be called dynamically, but the appropriate ADALNK (ADALNI) must be linked to MPPTB beforehand.

Linkage

Link your application to member MPPETB from the EntireX load library (EXX107.LOAD) and the appropriate Adabas link module from the Adabas load library.

NATETB23

NATETB23 is the recommended stub for Natural. See also *NATETBZ*.

Prerequisites and Installation Notes for Natural

- Set the Natural size parameters so that Natural can provide the stub with 34 KB at runtime.
- Send/receive buffers of greater than 32 KB can be used with NATETB23 provided that Adabas library WAL842 (or above) is installed. See *Contents of Mainframe Installation Medium* in the z/OS Installation documentation.

Linkage under Natural

- Linking the stub for use in z/OS Batch, TSO, CICS, IMS(BMP), IMS(MPP) and Com-plete:

  Link NATETB23 in the Load Library to the shared part of the Natural nucleus. NATETB23 is a reentrant and relocatable module.
- Alternatively, the NATETB23 stub can be dynamically invoked by the following Natural parameters:

```
RCA=BROKER,RCALIAS=(BROKER,NATETB23)
```

Installation Verification under Natural

To verify the installation of the stub under Natural

1 Log on to Natural library SYSRPC and type MENU.

2 Invoke SM Service Directory Maintenance from the main menu.

3 Define the Node and Server and save.

4 Invoke XC Server Command Execution from the main menu for the node and server defined in the previous step.

5 Ping the server with the command PI.

Your environment and the stub are installed correctly if you receive

- `02150148 Connection error`, meaning that the broker and the RPC server are down;

- `00070007 Service not registered`, meaning that the broker is up and the RPC server is down;
- an answer from the RPC server.

For other return codes, see *Error Messages and Codes*.

NATETBZ

This reentrant stub is for exclusive use by Natural RPC Server.

Prerequisites and Installation Notes for Natural

- It may run in SRB mode and is zIIP-eligible.
- Set the Natural size parameters so that Natural can provide the stub with 34 KB at runtime.

Linkage under Natural

- To *statically* link the stub for use in z/OS Batch, link NATETBZ into the load library where the Batch Natural nucleus resides. NATETBZ is a reentrant and relocatable module.
- Alternatively, the NATETBZ stub can be called *dynamically* with the following Natural parameters:

```
RCA=BROKER,RCALIAS=(BROKER,NATETBZ)
```

Tracing for Broker Stubs

Setting Trace

How to Set Trace	Stub Module				
	BROKER, MPPETB	**CICSETB**	**NATETB23**	**COMETB**	**NATETBZ, ARFETB, IDMSETB**
	Define the following DD statements in your JCL [(1)]:	Define the following DD statements in your JCL to start CICS [(2)]:	Define the following DD statements in your JCL [(3)]:	Add the following variables to the *EXAENV Environment Store* [(4)]:	Trace is not supported.

How to Set Trace			Stub Module				
			BROKER, MPPETB	CICSETB	NATETB23	COMETB	NATETBZ, ARFETB, IDMSETB
1	STANDARD	Traces initialization, errors, and all ACI request/reply strings.	`//EXALOG1 ↩ DD SYSOUT=* //TRACE1 ↩ DD SYSOUT=*`	`//EXALOG1 ↩ DD DUMMY`	`//EXALOG1 ↩ DD DUMMY //TRACE1 ↩ DD ↩ SYSOUT=*`	`EXA_LOG=1 ETB_STUBLOG=1`	
2	ADVANCED	Used primarily by system engineers, traces everything from level 1 and provides additional information, for example the Broker ACI control block, as well as information from the transports.	`//EXALOG2 ↩ DD SYSOUT=* //TRACE1 ↩ DD SYSOUT=*`	`//EXALOG2 ↩ DD DUMMY`	`//EXALOG2 ↩ DD DUMMY //TRACE1 ↩ DD ↩ SYSOUT=*`	`EXA_LOG=2 ETB_STUBLOG=2`	
3	SUPPORT	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.	`//EXALOG3 ↩ DD SYSOUT=* //TRACE1 ↩ DD SYSOUT=*`	`//EXALOG3 ↩ DD DUMMY`	`//EXALOG3 ↩ DD DUMMY //TRACE1 ↩ DD ↩ SYSOUT=*`	`EXA_LOG=3 ETB_STUBLOG=3`	
0	NONE	No tracing. Switch tracing off.	Do not define `EXALOG1`, `EXALOG2` and `EXALOG3` DD statements in your JCL.	Do not define `EXALOG1`, `EXALOG2` and `EXALOG3` DD statements in your JCL to start CICS.	Do not define `EXALOG1`, `EXALOG2` and `EXALOG3` DD statements in your JCL.	Do not define the variable `ETB_STUBLOG` and `EXA_LOG` in EXAENV environment store or set its value to zero.	

How to Set Trace	Stub Module				
	BROKER, MPPETB	CICSETB	NATETB23	COMETB	NATETBZ, ARFETB, IDMSETB
Trace Output Location	TCP trace output is written to TRACE1 DD statement while NET trace output is routed to EXALOG*n* DD statement defined in your JCL.	TCP as well as NET trace output is written to CICS TD queue CSSL. This TD queue is typically assigned to DD statement MSGUSER. Refer to the JCL to start CICS.	TCP trace output is written to TRACE1 DD statement	TCP trace output is controlled by the hardcopy setting in Com-plete. See hardcopy function from the UUTIL menu. If no device name has been defined, the trace is displayed on the terminal. If HC has been specified, the trace will be written to the Com-plete spool and routed to the device name supplied using HC=name. NET trace output is written via COMPLETE Printout Spooling, default destination "SYSOUT=X".	

Notes:

1. In this approach, DD assignments are checked by the stub module in order to determine the trace level and output location.
2. In this approach, DD assignments are checked by the stub module in order to determine the trace level.
3. NATETB23 supports TCP trace but does not support NET trace. For error detection with NET transport, see ***Tracing Broker Communication in Natural***.
4. In this approach a variable definition in a partitioned data set (PDS) member, the so-called EXAENV environment store is checked in order to determine the trace level. See ***EXAENV Environment Store***.

Examples

To set trace level 3 for stub module `BROKER` or `MPPETB`

- Define the following DD statements in your JCL:

```
//EXALOG3 DD SYSOUT=*
//TRACE1 DD SYSOUT=*
```

To set trace level 2 for stub module `CICSETB`

- Define the following DD statement in the JCL to start CICS:

```
//EXALOG2 DD DUMMY
```

To set trace level 1 for stub module `COMETB`

- Add the following variable name and value to the EXAENV environment store:

```
EXA_LOG=1
ETB_STUBLOG=1
```

Tracing Broker Communication in Natural

If a stub module for Natural does not support tracing (for example `NATETBZ`, `NATETB23` for NET transport) and you want detect errors in broker communication, we recommend using temporarily the platform-specific stub instead.

- For CICS, use

```
RCA=BROKER,RCALIAS=(BROKER,CICSETB)
```

- For Com-plete, use

```
RCA=BROKER,RCALIAS=(BROKER,COMETB)
```

  Also, PRIVILEG status is required in Com-plete for COMETB.

See *Available Stubs* for an overview of stub modules and ***Setting Trace***.

Timeout Settings for Broker Stubs

Setting timeouts for the transport layer is possible for the transport method TCP. The transport method NET does not support timeouts.

Setting the TCP Timeout

If the transport layer is interrupted, communication between the broker and the stub - that is, client or server application - is no longer possible. A client or server might possibly wait infinitely for a broker reply or message in such a situation. To prevent this and return control to your calling application in such a situation, set a timeout value for the transport method.

The timeout value for the transport method is set by the environment variable `ETB_TIMEOUT` on the stub side. This transport timeout is used together with the broker timeout - which is set by the application in the `WAIT` field of the broker ACI control block - to calculate the actual value for the transport layer's timeout. This timeout for the transport layer is independent of the timeout settings of the broker kernel.

The following table describes the possible values for the TCP timeout and how to set it per stub module:

How to Set the TCP Timeout		Stub Module		
		BROKER[2], MPPETB[2], BROKER[2], MPPETB[2], NATETB23[2], NATETBZ[3], ARFETB[3]	**COMETB[2]**	**IDMSETB**
		Use the *SAGTOKEN Utility* to set the TCP timeout.	Add the following variable to the *EXAENV Environment Store*:	TCP timeout is not supported.
0	Infinite wait for the application.	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('SET ↩ LOCAL,TIMEOUT=0') //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	`ETB_TIMEOUT=0`	

How to Set the TCP Timeout		Stub Module		
		BROKER[2], MPPETB[2], BROKER[2], MPPETB[2], NATETB23[2], NATETBZ[3], ARFETB[3]	COMETB[2]	IDMSETB
n	The transport method additionally waits this time in seconds. A negative value is treated as `TIMEOUT=0` (infinite wait for the application).[1]	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('SET ↩ LOCAL,TIMEOUT=n') //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	`ETB_TIMEOUT=n`	
nothing set	Transport method waits additional 20 seconds. [1]	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('DELETE ↩ LOCAL,TIMEOUT') //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	Do not define the variable `ETB_TIMEOUT` in EXAENV environment store.	

Notes:

1. The actual timeout for transport layer equals broker timeout (`WAIT` field) + transport timeout.
2. The stub supports this timeout in the connection phase and during data transfer.
3. The stub supports this timeout in the connection phase only.

Limiting the TCP Connection Lifetime

With transport method TCP/IP the broker stub establishes one or more TCP/IP connections to the brokers specified with `BROKER-ID`. These connections can be controlled by the transport-specific `CONNECTION-NONACT` attribute on the broker side, but also by the transport-specific environment variable `ETB_NONACT` on the stub side. If `ETB_NONACT` is not 0, it defines the non-activity time (in seconds) of active TCP/IP connections to any broker. See `ETB_NONACT` under *Environment Variables in EntireX*. Whenever the broker stub is called, it checks for the elapsed non-activity time and closes connections with a non-activity time greater than the value defined with `ETB_NONACT`. This timeout for the transport layer is independent of the timeout settings of the broker kernel.

The following table describes the possible values to limit TCP connection lifetime and how to set it per stub module:

How to Limit the TCP Connection Lifetime		Stub Module		
		BROKER, MPPETB, NATETB23	COMETB	NATETBZ[1], ARFETB[1], IDMSETB[1]
		Use the *SAGTOKEN Utility* to set the TCP timeout.	Add the following variable to the *EXAENV Environment Store*:	TCP connection lifetime cannot be limited.
0	Infinite wait for the application.	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('SET ↩ LOCAL,NONACT=0') //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	`ETB_NONACT=0`	
n	The transport method additionally waits this time in seconds. A negative value is treated as `TIMEOUT=0` (infinite wait for the application).[1]	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('SET ↩ LOCAL,NONACT=n') //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	`ETB_NONACT=n`	
nothing set	Transport method waits additional 20 seconds. [1]	`//STEP EXEC ↩ PGM=SAGTOKEN,PARM=('DELETE ↩ LOCAL,NONACT) //STEPLIB DD ↩ DISP=SHR,DSN=EXX107.LOAD`	Do not define the variable `ETB_NONACT` in EXAENV environment store.	

SAGTOKEN Utility

`SAGTOKEN` allows you to set variables. When setting variables with `SAGTOKEN`, `SAGTOKEN` error messages may be displayed on the operator console. The steplib EXX107.LOAD of `SAGTOKEN` must be APF-authorized. See *EntireX SAGTOKEN Messages*.

Syntax

```
//STEP EXEC PGM=SAGTOKEN,PARM=('command scope, variable=value')
//STEPLIB DD DISP=SHR,DSN=EXX107.LOAD
```

Operands

Command	Use
SET	Set or replace a SAGTOKEN variable.
DELETE	Delete a SAGTOKEN variable.
DISPLAY	Display a SAGTOKEN variable.

Scope	Meaning
LOCAL	Applies to the address space.
GLOBAL	Applies to the z/OS image.

Variable	Value
TIMEOUT	For information on TIMEOUT values, see *Timeout Settings for Broker Stubs*.
NONACT	For information on NONACT, see *Limiting the TCP Connection Lifetime*.

Note: If a job uses SAGTOKEN to set local tokens in one step, we recommend that you delete these tokens prior to job termination in order to release all acquired resources.

Transport Methods for Broker Stubs

Transport Method Values

Transport Value	Description / Tips
NET	Use Adabas Cross-Memory Services as transport method. See *Installing Adabas Components for EntireX under z/OS*. It is also possible to communicate remotely with the transport method NET from an application (client or server) to the broker kernel using Entire Net-Work. For remote NET communication, Entire Net-Work must be installed both on the machine where the broker kernel runs and on the machine where your application (client or server) runs, and a connection must be established. Using Adabas modules WAL811 or above allows more than 32 KB of data to be communicated. Otherwise the following maximum values are allowed:

<table>
<tr><th>Transport Value</th><th>Description / Tips</th></tr>
<tr><td></td><td>
<table>
<tr><th>ACI Version</th><th>Max Send/Receive length</th></tr>
<tr><td>1</td><td>32167</td></tr>
<tr><td>2, 3</td><td>31647</td></tr>
<tr><td>4-8</td><td>31643</td></tr>
<tr><td>9 or above</td><td>31123</td></tr>
</table>
Note: If Adabas version 8 or above is not used, these same limits still apply under z/OS.
</td></tr>
<tr><td>TCP</td><td>Use TCP/IP as transport method.</td></tr>
</table>

For Secure Sockets Layer/Transport Layer Security (SSL/TLS) as transport method, see table *Using SSL/TLS with EntireX Components*.

Setting the Default Transport Method

For stub module BROKER, DD assignments are checked in order to determine the default transport method.

Transport	How to set Default Transport?
TCP	Define `EXATCP` as dummy DD statement in your JCL: `//EXATCP DD DUMMY`
NET	Define `EXANET` as dummy DD statement in your JCL: `//EXANET DD DUMMY`

SVC Number for Broker Communication

Stub	Notes
ARFETB	See your Event Replicator for Adabas documentation for information on how to specify the SVC number used for Broker communication.
BROKER	When Entire Net-Work transport is used, the default SVC number (249) can be overridden in the following ways: ■ By specifying the SVC number as part of the Broker ID, for example:

Stub	Notes
	`ETB220:SVC237:NET` ■ By including ADAUSER when linking the stub with the application. This enables the SVC number to be specified in the ADARUN cards of the application job. This option cannot be employed if the application operates within a multithreading application (multiple TCBs). ■ By using the supplied zap if neither of the above options is chosen. See `BROKER`.
CICSETB	When Entire Net-Work transport is used and the default SVC number (249) has to be changed, the Adabas communications module under CICS determines how the SVC number can be changed. See the Adabas documentation for information on how to change the SVC number.
COMETB	■ The Adabas interface is integrated within the TP Monitor Com-plete. ■ Add the `ADASVC5` parameter to Com-plete (or Adabas TPF) startup parameters as specified below: `ADASVC5=(dbid,svc)` where *dbid* is the node ID selected for use by the EntireX Broker address space, and *svc* is the Adabas SVC number used by the EntireX Broker started task/job. ■ Optionally consider how to specify the SVC number used for Broker communication maintained within Com-plete's Adabas interface. See the Com-plete documentation.
MPPETB	When Entire Net-Work transport is used and the default SVC number (249) has to be changed, the Adabas communications module under IMS/DC determines how the SVC number can be changed. See the Adabas documentation for information on how to change the SVC number.
NATETB23	■ If you are communicating with EntireX Broker from Com-plete or Adabas TPF, add the `ADASVC5` parameter to Com-plete (or Adabas TPF) startup parameters as specified below: `ADASVC5=(dbid,svc)` where *dbid* is the node ID selected for use by the EntireX Broker address space, and *svc* is the Adabas SVC number used by the EntireX Broker started task/job. ■ For all other environments, see the Natural documentation for information on how to specify the SVC number used for Broker communication.
NATETBZ	See your Natural documentation for information on how to specify the SVC number used for Broker communication.

Considerations for Users without Adabas

For customers who do not have Adabas installed at their site,

- we recommend installing the Adabas modules (library WAL842) delivered with EntireX. See also *Contents of Mainframe Installation Medium* in the z/OS Installation documentation.
- the Adabas modules will greatly improve performance if the transport method NET is used and broker kernel and applications (client or server) communicating through the stub to the broker kernel on the same machine locally, see ***Transport Methods for Broker Stubs***.

For information on how to install the Adabas SVC and install Adabas with TP Monitors, see *Installing Adabas Components for EntireX under z/OS*.

Support of Clustering in a High Availability Scenario

EntireX Broker supports clustering in a high-availability scenario, using the environment variable `ETB_SOCKETPOOL`. See *Environment Variables in EntireX*. This section covers the following topics:

See also *High Availability in EntireX*.

Introduction

A TCP/IP connection established between stub and broker is not exclusively assigned to a particular thread. With multithreaded applications, two or more threads may use the same connection. On the other hand, if a connection is busy, another new one is created to exchange data.

In order to access the same broker instance in a clustering environment, an affinity between application thread and TCP/IP connection is needed to always use the same connection within an application thread. Therefore, an environment variable is evaluated to control the handling of TCP/IP connections.

If environment variable `ETB_SOCKETPOOL` is set to "OFF" (`ETB_SOCKETPOOL=OFF`), an affinity between threads and TCP/IP connections is established. All requests to one particular broker will use the same TCP/IP connection. `ETB_SOCKETPOOL` controls all TCP/IP connections.

Stubs `ARFETB` and *`NATETBZ`* always establish an affinity between subtask and TCP/IP connection.

Exceptions

Broker attribute `CONNECTION-NONACT` is used by the broker to close TCP/IP connections after the elapsed non-activity time. Omit this attribute to keep the TCP/IP connection alive.

Default

`ETB_SOCKETPOOL=ON` is the default setting. In this case, an established broker connection can be used by any thread if the connection is not busy.

Restriction

Support for this feature is currently not available under CICS, Com-plete, and IDMS/DC.

8 Broker Command-line Utilities

EntireX Broker provides the following internal services: Command Service and Information Service, which can be used to administer and monitor brokers. Because these services are implemented internally, nothing has to be started or configured. You can use these services immediately after starting EntireX Broker.

ETBINFO

Queries the Broker for different types of information, generating an output text string with basic formatting. This text output can be further processed by script languages. `ETBINFO` uses data descriptions called profiles to control the type of data that is returned for a request. `ETBINFO` is useful for monitoring and administering EntireX Broker efficiently, for example how many users can run concurrently and whether the number of specified message containers is large enough.

Although basic formatting of the output is available, it is usually formatted by script languages or other means external to the Broker.

Running the Command-line Utility

In a z/OS environment, run the command-line utility `ETBINFO` as shown below.

Note: The service data set name and member cannot be accessed directly, but only indirectly by its DD name in the JCL.

```
//ETBINFO EXEC PGM=ETBINFO,
// PARM=('ENVAR(''ETB_STUBLOG=0'')/ -p "//''EXX107.JOBS(BROKER)''" ',
//       '-bbrokerid -dBROKER -xuid -ypwd')
//STEPLIB  DD   DISP=SHR,DSN=< EXB-load-lib >
//         DD   DISP=SHR,DSN=< EXX-load-lib >
//         DD   DISP=SHR,DSN=< WAL-load-lib >
//TRACE1   DD   SYSOUT=*                              stublog
//SYSOUT   DD   SYSOUT=*                              stderr
//SYSPRINT DD   SYSOUT=*                              stdout
//
```

The SVC number specified in the `ADARUN` parameter must match the SVC number used for the target Broker ID.

Command-line Parameters

The table below explains the command-line parameters. The format string and profile parameters are described in detail following the table. All entries in the Option column are case-sensitive.

Option	Command-line Parameter	Req/ Opt	Explanation
-b	brokerid	R	Broker identifier, for example localhost:1971:TCP.
-c	class	O	Class as selection criterion.
-C		O	Create output with comma-separated values, suitable for input into a spreadsheet or other analysis tool. Any format string specified by means of format string or profile command-line parameters is ignored.
-d	object	R	Possible values: Object — Provides Info on BROKER — Broker. CLIENT — Client. CMDLOG-FILTER — Command log filter. CONVERSATION — Conversation. NET — Entire Net-Work transport. PARTICIPANT — Participant. POOL-USAGE — Broker pool usage. PSF — Unit-of-work status. PSFADA — Adabas persistent store. PSFCTREE — c-tree persistent store. PSFDIV — DIV persistent store. RESOURCE-USAGE — Broker resource usage. SECURITY — EntireX Security. SERVER — Server. SERVICE — Service. SSL — SSL transport. STATISTICS — Broker statistics. TCP — TCP transport. UOW-STATISTICS — Units of work per service. USER — Participant (short). WORKER — Worker. WORKER-USAGE — Worker usage.
-e	recv class	O	Receiver's class name. This selection criterion is valid only for object PSF.

Option	Command-line Parameter	Req/ Opt	Explanation
-f	*Format String*	O	Format string how you expect the output. See *Profile*.
-g	recv service	O	Receiver's service name. This selection criterion is valid only for object PSF.
-h	help	O	Prints help information.
-i	convid	O	Conversation ID as selection criterion. Only valid for object CONVERSATION.
-I	conv type	O	Conversation's type.
-j	recv server	O	Receiver's server name. This selection criterion is valid only for object PSF.
-k	recv token	O	Receiver's token. This selection criterion is valid only for object PSF.
-l	level	O	The amount of information displayed: FULL All information. SHORT User-specific information.
-m	recv userid	O	Receiver's user ID. This selection criterion is valid only for object PSF.
-n	server name	O	Server name. This selection criterion is valid only for the objects SERVER, SERVICE or CONVERSATION.
-p	pds(member)	O	Here you can specify a PDS member that defines the layout of the output. There are default files you can modify or you can use your own. The default files are: BROKER CLIENT CLOGFLT CONV NET POOL PSF PSFADA PSFCTREE PSFDIV SERVICE SSL STATIS STATIS TCP USER WORKER WKRUSAGE See *Profile*.
-q	puserid	O	Physical user ID. This selection criterion is valid only for objects CLIENT, SERVER, CONVERSATION, **Note:** Must be a hex value.
-r	sec	O	Refresh information after seconds.
-s	service	O	Service. This selection criterion is valid only for objects SERVER, SERVICE or CONVERSATION.
-t	token	O	This selection criterion is valid only for objects CLIENT, SERVER, SERVICE or CONVERSATION.
-u	userid	O	User ID. This selection criterion is only valid for the display types CLIENT, SERVER, SERVICE or CONVERSATION.
-v	UOW status	O	Unit of work status. This selection criterion is valid only for object PSF.
-w	UOW ID	O	Unit of work ID. This selection criterion is valid only for object PSF.

Option	Command-line Parameter	Req/ Opt	Explanation
-x	userid	O	User ID. For security purposes.
-y	password	O	Password. For security purposes.
-z	token	O	Used with userid to uniquely identify a caller to Command and Information Services.

Command-line Parameters from File

ETBINFO supports an alternative method of passing command-line parameters.

If the DDNAME INFFILE is allocated, using

```
//INFFILE  DD  DISP=SHR,DSN=pds(member)
```

and no command-line parameters are specified in the EXEC instruction, the content of the allocated member is evaluated. See sample member below (the apostrophes are included to show the record format (LRECL 80):

```
'-blocalhost:3930:TCP                                                          '
'-dBROKER                                                                      '
```

If ETBINFO is configured using INFFILE and a profile is specified, the syntax of the -p is as follows:

-p//'dsname(member)'	if a PDS member is used
-p//'dsname'	if a sequential data set is used

The syntax is based on IBM conventions for fopen.

Caution: Make sure INFFILE does not contain line numbers in columns 73-80. If line numbers are present, arguments are not passed correctly. Only one option per line is supported.

Profile

If you do not use the profile option or a format string, your output will be an unformatted list with all columns of that display type. To display specific columns, specify a profile that includes only those columns.

The following default sample profiles include all the columns defined for each display type:

- BROKER
- CLIENT
- CLOGFLT
- CONV
- NET
- POOL
- PSF
- PSFADA
- PSFCTREE
- PSFDIV
- RESOURCE
- SECURITY
- SERVER
- SERVICE
- SSL
- STATIS
- TCP
- USER
- WKRUSAGE
- WORKER

You can either delete the columns not required or copy the default profile and modify the order of the columns. Ensure that the column names have a leading "%". Column names can be written in one line or on separate lines. The output is always written side by side. With profile parameters `%DATE` and `%TIME` you can provide a timestamp for the command-line query.

Location of Profiles

On z/OS, the profiles used to control the format of data displayed are members of the EXB source library and are named `SERVER`, `CLIENT`, etc. They can be saved as either sequential files or PDS members.

Example 1

Profile for object SERVICE: SERVICE.

```
//ETBINFO EXEC PGM=ETBINFO,PARM=('/-b ETB001::NET -d SERVICE ',
//         '-p "//''EXX107.JOBS(SERVICE)''" -l FULL -xUID')
```

The following list is displayed:

```
SAG                                ETBCIS     INFO
 1 0 16 86400 0 31647 0 00 00 00 00 0 0
SAG                                ETBCIS     USER-INFO
 2 0 16 86400 0 31647 0 00 00 00 00 0 0
SAG                                ETBCIS     CMD
 6 0 16 86400 0 31647 0 00 00 00 00 0 0
```

Example 2

Your own profile: MYPROF

```
//ETBINFO EXEC PGM=ETBINFO,PARM=('/-b ETB001::NET -d SERVICE ',
//         '-p "//''EXX107.JOBS(MYPROF)''" -xUID')
```

Note: In this case, `MYPROF` contains:`%4.4SERVERCLASS %SERVERNAME`

The following list is displayed:

```
ACLA    ASERVER
BCLA    BSERVER
CCLA    CSERVER
```

Sample Profiles for ETBINFO

You can find the sample profiles for `ETBINFO` in your EXB source library.

Format String

The format string, if specified, will override the use of a profile. The format string is built like a `printf()` in C language. The string must be enclosed in quotation marks. You can specify the columns by using a "%" and the column name. The column name must contain letters only. Numeric characters are not allowed. You can specify the length of column output by using a format precision, as in the ANSI-C `printf()` function. The column name must be followed by a blank. For example:

```
etbinfo -b ETB001 -d BROKER -f "%12.12CPLATNAME   %NUM-SERVER      %NUM-CLIENT"
```

which produces the following output, for example:

```
MVS/SP 7.04 30 100
```

You can also use an arbitrary column separator, which can be any character other than "%". You can use `\n` for a new line in the output and `\t` for a tabulator in the format string or profile. For example:

```
etbinfo -b ETB001 -d SERVER -f "UserID: %5.5USER-ID Token: %5.5TOKEN"
```

which produces:

```
UserID: HUGO  Token: MYTOK
UserID: EGON  Token:
UserID: HELMU Token: Helmu
```

If you want to structure your output a little more, you can operate with the `\n` or `\t` character. For example:

```
etbinfo -b ETB001 -d SERVICE -f  "Class:%5.5SERVER-CLASS \n\tName:%5.5SERVER-NAME ↩
\n\tService:%5.5SERVICE"
```

which produces:

```
Class:DATAB
    Name:DB10
    Service:Admin
Class:PRINT
    Name:LPT1
    Service:PRINT
...
```

You can also add a timestamp to the query:

```
etbinfo -b ETB001 -d BROKER -f "%DATE %TIME"
```

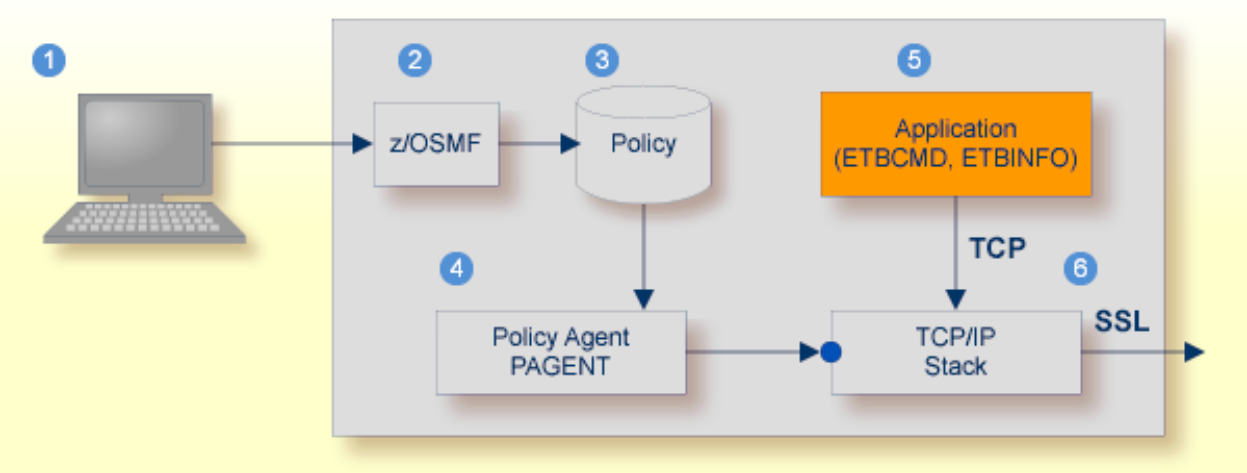
which produces:

```
2014-08-19 10:00:00.234
```

Using SSL/TLS

For establishing an SSL connection on z/OS, IBM's Application Transparent Transport Layer Security (AT-TLS) can be used where the establishment of the SSL connection is pushed down the stack into the TCP layer.

Configure the AT-TLS rules for the policy agent (`PAGENT`) 4 using an appropriate client 1 and the z/OS Management Facility (z/OSMF) 2. Together with SSL parameters (to provide certificates stored in z/OS as RACF keyrings) define AT-TLS rules, for example by using the application 5 job name and local TCP port number. If the rules match, the TCP connection is turned into an SSL connection 6. Refer to your IBM documentation for more information, for example the IBM Redbook *Communications Server for z/OS VxRy TCP/IP Implementation Volume 4: Security and Policy-Based Networking*.



1. Client to interact with z/OS Management Facility (z/OSMF).
2. AT-TLS rules are defined with z/OSMF policy management.
3. Policy Repository with AT-TLS rules stored as z/OS files.
4. Policy Agent, MVS task `PAGENT`, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.
5. Application using TCP connection.
6. If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

1. The client 1 may vary per operating system, for example a Web browser for z/OS 2.1.
2. z/OSMF 2 includes other administration and management tasks in addition to policy management.
3. Policy Management 3 includes other rules, such as IP filtering, network address translation etc.

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Set up the tool for a TCP/IP connection. On mainframe platforms, use *Transport-method-style Broker ID*. Example:

```
ETB024:1699:TCP
```

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

4 Make sure the broker is prepared for SSL connections as well. See *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation.

ETBCMD

Allows the user to take actions - for example purge a unit of work, stop a server, shut down a Broker - against EntireX Broker.

Running the Command-line Utility

In a z/OS environment, run the `ETBCMD` command-line utility like this:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('ENVAR(''ETB_STUBLOG=0'')/ -bbrokerid ',
//       '-dBROKER -c... -xuid -ypwd')
//STEPLIB  DD   DISP=SHR,DSN=< EXB-load-lib >
//         DD   DISP=SHR,DSN=< EXX-load-lib >
//         DD   DISP=SHR,DSN=< WAL-load-lib >
//TRACE1   DD   SYSOUT=*                              stublog
//SYSOUT   DD   SYSOUT=*                              stderr
//SYSPRINT DD   SYSOUT=*                              stdout
//
```

The SVC number specified in the ADARUN parameter must match the SVC number used for the target Broker ID.

Command-line Parameters

The table below explains the command-line parameters. All entries in the **Option** column are case-sensitive.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
`brokerid`	`-b`	e.g. `ETB001`	R	Broker ID.
`command`	`-c`	■ `ALLOW-NEWUOWMSGS` ■ `APPMON-ON` ■ `APPMON-OFF` ■ `CLEAR-CMDLOG-FILTER` ■ `CONNECT-PSTORE`	R	Command to be performed. See *List of Commands and Objects* below.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
		■ DISABLE-ACCOUNTING ■ DISABLE-CMDLOG-FILTER ■ DISABLE-CMDLOG ■ DISABLE-DYN-WORKER ■ DISCONNECT-PSTORE ■ ENABLE-ACCOUNTING ■ ENABLE-CMDLOG-FILTER ■ ENABLE-CMDLOG ■ ENABLE-DYN-WORKER ■ FORBID-NEWUOWMSGS ■ PING ■ PRODUCE-STATISTICS ■ PURGE ■ RESET-USER ■ RESUME ■ SET-CMDLOG-FILTER ■ SET-COLLECTOR ■ SET-UOW-STATUS ■ SHUTDOWN ■ START ■ STATUS ■ STOP ■ SUSPEND ■ SWITCH-CMDLOG ■ TRACE-FLUSH ■ TRACE-OFF ■ TRACE-ON ■ TRAP-ERROR		
object type	-d	■ BROKER ■ CONVERSATION ■ PARTICIPANT ■ PSF ■ SECURITY	R	The object type to be operated on. See *List of Commands and Objects* below. Within EntireX Broker nomenclature, a participant is an application implicitly or explicitly logged on to the Broker as a specific user. See *Implicit Logon* and *Explicit*

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
		■ SERVER ■ SERVICE ■ TRANSPORT		*Logon*. A participant could act as client or server.
	-D	collector brokerid	O	For command SET-COLLECTOR only. If provided, sets the collector ID to the given collector broker ID.
	-e	errornumber	O	Error number being trapped.
	-E		O	Exclude attach servers from service shutdown.
help	-h		O	Prints help information.
class/server/service	-n	class/server/service	O	Service triplet.
option	-o	■ ACCEPTED ■ CANCELLED ■ IMMED ■ QUIESCE ■ LEVEL*n*, where *n*=1-8	O	Command option.
puserid	-p	puserid	O	Physical User ID. For SERVER and PARTICIPANT objects only. This must be a hex value.
seqno	-S	sequence number	O	Sequence number of participant.
token	-t	token	O	Token. For PARTICIPANT object only.
uowid	-u	uowid	O	Unit of work ID. For PSF object only.
userid	-U	userid	O	User ID. For PARTICIPANT object only.
secuserid	-x	userid	O	User ID for security purposes.
transportid	-X	Transport ID	O	One of the following: COM|NET|SSL|S*nn*|TCP|T*nn*. See table below.
secpassword	-y	password	O	Password for security purposes.

Transport ID Values

This table explains the possible values for parameter transportid:

Transport ID	Explanation
COM	all communicators
NET	NET transport communicator
SSL	all SSL communicators
S00	SSL communicator 1
S01	SSL communicator 2
S02	SSL communicator 3
S03	SSL communicator 4
S04	SSL communicator 5
TCP	all TCP/IP communicators
T00	TCP/IP communicator 1
T01	TCP/IP communicator 2
T02	TCP/IP communicator 3
T03	TCP/IP communicator 4
T04	TCP/IP communicator 5

Command-line Parameters from File

ETBCMD supports an alternative method of passing command-line parameters.

If the DDNAME CMDFILE is allocated, using

```
//CMDFILE  DD  DISP=SHR,DSN=pds(member)
```

and no command-line parameters are specified in the EXEC instruction, the content of the allocated member is evaluated. See sample member below (the apostrophes are included to show the record format (LRECL 80):

```
'-blocalhost:3930:TCP                                                    '
'-dBROKER                                                                '
```

Caution: Make sure CMDFILE does not contain line numbers in columns 73-80. If line numbers are present, arguments are not passed correctly. Only one option per line is supported.

List of Commands and Objects

This table lists the available commands and the objects to which they can be applied.

Command	Object							
	BROKER	CONVERS-ATION	PARTICI-PANT	PSF	SECURITY	SERVER	SERVICE	TRANSPORT
ALLOW-NEWUOWMSGS				x				
APPMON-OFF	x							
APPMON-ON	x							
CLEAR-CMDLOG-FILTER	x							
CONNECT-PSTORE				x				
DISABLE-ACCOUNTING	x							
DISABLE-CMDLOG-FILTER	x							
DISABLE-CMDLOG	x							
DISABLE-DYN-WORKER	x							
DISCONNECT-PSTORE				x				
ENABLE-ACCOUNTING	x							
ENABLE-CMDLOG-FILTER	x							
ENABLE-CMDLOG	x							
ENABLE-DYN-WORKER	x							
FORBID-NEWUOWMSGS				x				

Command	Object							
	BROKER	CONVERS-ATION	PARTICI-PANT	PSF	SECURITY	SERVER	SERVICE	TRANSPORT
PING	x							
PRODUCE-STATISTICS	x							
PURGE				x				
RESET-USER					x			
RESUME								x
SET-CMDLOG-FILTER	x							
SET-COLLECTOR	x							
SET-UOW-STATUS				x				
SHUTDOWN	x	x	x			x	x	
START								x
STATUS								x
STOP								x
SUSPEND								x
SWITCH-CMDLOG	x							
TRACE-FLUSH	x							
TRACE-OFF	x			x	x			
TRACE-ON	x			x	x			
TRAP-ERROR	x							

Examples

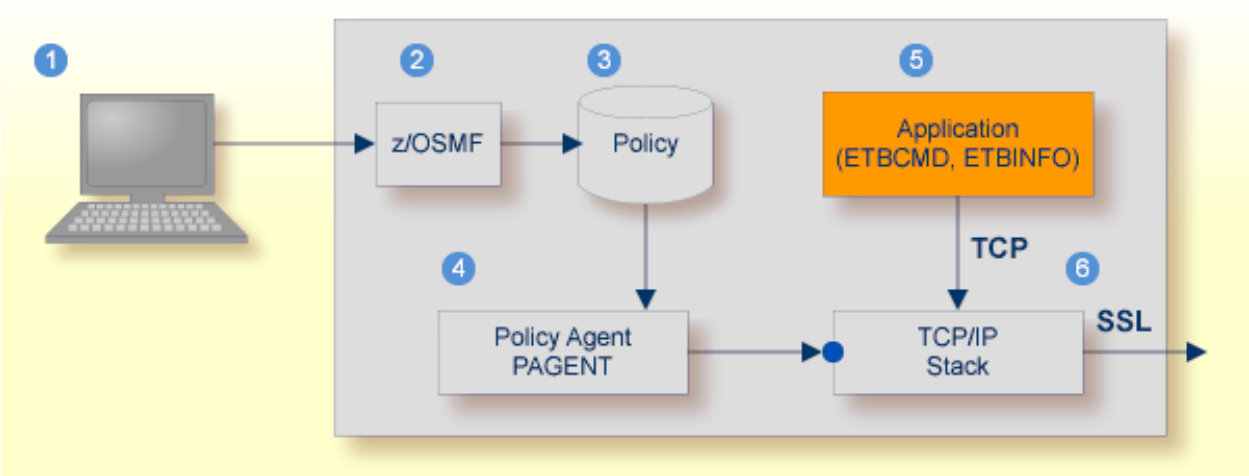
Example	Description
`etbcmd -b etb001 -h`	Displays ETBCMD help text.
`etbcmd -b etb001 -d BROKER -c TRACE-OFF`	Turns Broker tracing off.
`etbcmd -b etb001 -d BROKER -c TRACE-ON -o LEVEL2`	Sets Broker trace level to 2.
`etbcmd -b etb001 -d BROKER -c SHUTDOWN`	Performs Broker shutdown.
`etbcmd -b etb001 -d SERVICE -c SHUTDOWN -o IMMED -n ACLASS/ASERVER/ASERVICE`	Shut down service `CLASS=ACLASS,SERVER=ASERVER,SERVICE=ASERVICE`. See also `SHUTDOWN SERVICE` under *Broker Command and Information Services* in the EntireX Broker documentation for more information on shutdown options.
	Create list of servers and shutdown specific server in two steps (first step uses `ETBINFO`). See also `SHUTDOWN SERVER`.

Example	Description
`etbinfo -b etb001 -d SERVER -l FULL -f"%USER-ID %SEQNO"`	1. Determine a list of all servers with sequence numbers.
`etbcmd -b etb001 -d SERVER -c SHUTDOWN -o IMMED -S32`	2. Shutdown server with sequence number 32.
`etbcmd -b etb001 -d BROKER -c PING`	Performs an EntireX ping against the Broker.
`etbcmd -b etb001 -d PSF -c DISCONNECT-PSTORE`	Disconnects the Broker PSTORE.
`etbcmd -b etb001 -d PSF -c CONNECT-PSTORE`	Connects the Broker PSTORE.
`etbcmd -b etb001 -d PSF -c PURGE -u 10000000U00001A`	Purges a unit of work.
`etbcmd -b etb001 -d PSF -c ALLOW-NEWUOWMSGS`	Allows new units of work to be stored.
`etbcmd -b etb001 -d PSF -c FORBID-NEWUOWMSGS`	Disallows new units of work to be stored.
`etbcmd -b etb001 -d PSF -c SET-UOW-STATUS -o ACCEPTED -n ACLASS/ASERVER/ASERVICE`	Sets the status of UOWs that reside in the postpone queue back to `ACCEPTED` for service `ACLASS/ASERVER/ASERVICE`. See also *Postponing Units of Work* under *Using Persistence and Units of Work* in the platform-independent Administration documentation.
`etbcmd -b etb001 -d PSF -c SET-UOW-STATUS -o CANCELLED -u 0010000000000100`	Cancel UOW with UOWID 0010000000000100 that resides in the postpone queue. See also *Postponing Units of Work*.

Using SSL/TLS

For establishing an SSL connection on z/OS, IBM's Application Transparent Transport Layer Security (AT-TLS) can be used where the establishment of the SSL connection is pushed down the stack into the TCP layer.

Configure the AT-TLS rules for the policy agent (`PAGENT`) 4 using an appropriate client 1 and the z/OS Management Facility (z/OSMF) 2. Together with SSL parameters (to provide certificates stored in z/OS as RACF keyrings) define AT-TLS rules, for example by using the application 5 job name and local TCP port number. If the rules match, the TCP connection is turned into an SSL connection 6. Refer to your IBM documentation for more information, for example the IBM Redbook *Communications Server for z/OS VxRy TCP/IP Implementation Volume 4: Security and Policy-Based Networking*.



1. Client to interact with z/OS Management Facility (z/OSMF).
2. AT-TLS rules are defined with z/OSMF policy management.
3. Policy Repository with AT-TLS rules stored as z/OS files.
4. Policy Agent, MVS task `PAGENT`, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.
5. Application using TCP connection.
6. If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

1. The client 1 may vary per operating system, for example a Web browser for z/OS 2.1.
2. z/OSMF 2 includes other administration and management tasks in addition to policy management.
3. Policy Management 3 includes other rules, such as IP filtering, network address translation etc.

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Set up the tool for a TCP/IP connection. On mainframe platforms, use *Transport-method-style Broker ID*. Example:

```
ETB024:1699:TCP
```

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

4 Make sure the broker is prepared for SSL connections as well. See *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation.

9 Operator Commands

Command Syntax

The following command format is required to communicate with EntireX Broker, using the operator console. Parameters in UPPERCASE must be typed “as is”. Parameters in lowercase must be substituted with a valid value. Operator commands have the following format:

```
F task_name,command[parameter]
```

where *task_name* is the name of the EntireX Broker started task or job at your installation

command is the operator command

parameter is an optional parameter allowed by the operator command you are issuing

Note: In earlier versions, the command prefix `"APPL="` was required by an internal interface that is no longer used. You may omit `"APPL="` now, but this is still valid for compatibility reasons.

General Broker Commands

The following broker commands are available:

- BROKER TRACE
- DPOOL
- DRES
- DSTAT
- DWM
- ETBEND
- ETBSTOP
- FLUSH
- PSTORE TRACE
- SHUTDOWN class,server,service
- TRACE

- TRAP-ERROR

BROKER TRACE

Alias of broker command `TRACE`. Modifies the setting of the broker-specific attribute `TRACE-LEVEL`.

Example

To set a trace level 2 for broker

- Enter command:

```
/F taskname,BROKER TRACE=2
```

If the console prompt is suppressed, enter an `MSG` command before the console command:

```
MSG partition_id
```

See `TRACE-LEVEL` under *Broker-specific Broker Attributes*.

DPOOL

Lists all memory pools currently allocated by EntireX Broker. Start address, pool size in bytes and name of pool are provided. There can be multiple entries for a specific type of pool.

Sample Output

```
ETBM0720 Operator typed in: DPOOL
ETBM0657 Broker pool usage:
ETBM0657 0x2338FFB8   16781380 bytes COMMUNICATION POOL
ETBM0657 0x243A9EB8     368964 bytes CONVERSATION POOL
ETBM0657 0x24404F38     233668 bytes CONNECTION POOL
ETBM0657 0x2443EF38    4395204 bytes LONG MESSAGES POOL
ETBM0657 0x24870BB8    3703876 bytes SHORT MESSAGES POOL
ETBM0657 0x24BF9398     134244 bytes PARTICIPANT POOL
ETBM0657 0x24C1AF78      36996 bytes PARTICIPANT EXTENSION POOL
ETBM0657 0x24C24798      26724 bytes PROXY QUEUE POOL
ETBM0657 0x24C2BDA8     131668 bytes SERVICE ATTRIBUTES POOL
ETBM0657 0x24C4CB98      54372 bytes SERVICE POOL
ETBM0657 0x24C5AF78      32900 bytes SERVICE EXTENSION POOL
ETBM0657 0x24C63B18      87268 bytes TIMEOUT QUEUE POOL
ETBM0657 0x24C79398     179300 bytes TRANSLATION POOL
ETBM0657 0x24CA5F38     176324 bytes UNIT OF WORK POOL
ETBM0657 0x24CD1798     391268 bytes WORK QUEUE POOL
ETBM0582 Function completed
```

DRES

Displays EntireX Broker's resource usage for conversations, message buffers, participants, services, the timeout queue, units of work, and the work queue. Resource usage provides the total number, the number of free elements, and the number of used elements.

Sample Output

```
ETBM0720 Operator typed in: DRES
ETBM0581 Broker resource usage:
ETBM0581 Resource ----------------- Total # --- Free # --- Used #
ETBM0581 Conversations                 4096        852       3244
ETBM0581 Long message buffers             0          0          0
ETBM0581 Short message buffers         8192       7384        808
ETBM0581 Participants                   256        235         21
ETBM0581 Services                       256        240         16
ETBM0581 Timeout Queue                 1280        845        435
ETBM0581 Units Of Work                    0          0          0
ETBM0581 Work Queue                     256        239         17
ETBM0582 Function completed
```

DSTAT

Displays the total number of active elements, and an optional high watermark for services, clients, servers, conversations and message buffers.

Sample Output

```
ETBM0720 Operator typed in: DSTAT
ETBM0580 Broker statistics:
ETBM0580 NUM-SERVICE ............... 0
ETBM0580 Services active ........... 7
ETBM0580 NUM-CLIENT ................ 0
ETBM0580 Clients active ............ 10
ETBM0580 Clients active HWM ........ 10
ETBM0580 NUM-SERVER ................ 0
ETBM0580 Servers active ............ 10
ETBM0580 Servers active HWM ........ 10
ETBM0580 NUM-CONVERSATION .......... 0
ETBM0580 Conversations active ...... 607
ETBM0580 Conversations active HWM .. 968
ETBM0580 NUM-LONG-BUFFER ........... 0
ETBM0580 Long buffers active ....... 0
ETBM0580 Long buffers active HWM ... 0
ETBM0580 NUM-SHORT-BUFFER .......... 0
ETBM0580 Short buffers active ...... 1219
ETBM0580 Short buffers active HWM .. 1928
ETBM0582 Function completed
```

DWM

If broker attribute `DYNAMIC-WORKER-MANAGEMENT=YES` is activated, use command `DWM=OFF` to switch off dynamic worker management, or `DWM=ON` to reactivate it.

Example

To deactivate dynamic worker management

- Enter command:

```
/F taskname,DWM=OFF
```

See *Dynamic Worker Management*.

ETBEND

Processing stops immediately. Current calls to the EntireX Broker are not allowed to finish.

ETBSTOP

Alias of `ETBEND`.

FLUSH

Flush all trace data kept in internal trace buffers to stderr (`DD:SYSOUT`). The broker-specific attribute `TRMODE=WRAP` is required.

PSTORE TRACE

Modifies the trace level for the Adabas persistent store (Adabas-specific attribute `TRACE-LEVEL`).

Example

To set a trace level 2 for the Adabas persistent store

- Enter command:

```
/F taskname,PSTORE TRACE=2
```

See TRACE-LEVEL under *Adabas-specific Broker Attributes*.

SHUTDOWN class,server,service

Shuts down the specified service immediately and stops all servers that have registered this service.

Example

To shut down service CLASS=RPC, SERVER=SRV1, SERVICE=CALLNAT

- Enter command:

```
/F taskname,SHUTDOWN RPC,SRV1,CALLNAT
```

TRACE

Modifies the setting of the broker-specific attribute TRACE-LEVEL.

Sample Commands

To modify the trace level

- Enter command, for example:

```
/F taskname,TRACE=0
/F taskname,TRACE=1
/F taskname,TRACE=4
```

See TRACE-LEVEL under *Broker-specific Broker Attributes*.

TRAP-ERROR

Modifies the setting of the broker-specific attribute TRAP-ERROR.

Sample Command

To modify the setting for TRAP-ERROR

- Enter command:

```
/F taskname,TRAP-ERROR=nnnn
```

where *nnnn* is the four-digit API error number that triggers the trace handler.

See TRAP-ERROR under *Broker-specific Broker Attributes*.

Participant-specific Commands

Within EntireX Broker nomenclature, a participant is an application implicitly or explicitly logged on to the Broker as a specific user. See *Implicit Logon* and *Explicit Logon*. A participant could act as client or server. The following participant-specific commands are available:

- CANCEL parameter
- FREEZE
- RUN
- USERLIST
- USERS parameter

CANCEL parameter

Operator command `CANCEL` is used to delete participants from EntireX Broker. The following parameters are supported:

Parameter	Description
`[USER=]`*`user_id`*	Cancel all participants with the specified *`user_id`*. Non-persistent resources will be freed by the timeout manager. Prefix "`USER=`" is the default value and may be omitted.
`SEQNO=`*`seqno`*	Cancel the participant with the sequence number *`seqno`*. Non-persistent resources will be freed by the timeout manager. Operator commands `USERLIST` and `USERS` display sequence numbers of all selected participants.

Sample Commands

To cancel all participant entries of user "DOE"

- Enter command:

```
/F taskname,CANCEL DOE
```

Or:

```
/F taskname,CANCEL USER=DOE
```

To cancel participant with sequence number "11"

- Enter command:

```
/F taskname,CANCEL SEQNO=11
```

FREEZE

Operator command `FREEZE` freezes user request processing in Broker.

Sample Command

To freeze user request processing in Broker

- Enter command:

```
/F taskname,FREEZE
```

RUN

Operator command `RUN` resumes user request processing in Broker.

Sample Command

To resume user request processing in Broker

- Enter command:

```
/F taskname,RUN
```

USERLIST

Operator command `USERLIST` displays a list of selected participant entries. The following parameters are supported:

Parameter	Description
none | *	Display all participants.
user_id	Display all participants with user ID *user_id*. Wildcard characters are supported.

Sample Commands

To display all participants

- Enter command:

```
/F taskname,USERLIST
```

Or:

```
/F taskname,USERLIST *
```

To display all participants with user ID "DOE"

- Enter command:

```
/F taskname,USERLIST DOE
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST DOE
ETBM0687 Participants:
ETBM0687 USER-ID ----------------------- C S P U E CHR SEQNO
ETBM0687 DOE                             N Y N N Y ASC 1
ETBM0582 Function completed
```

To display all participants with user ID starting with uppercase "D"

- Enter command:

```
/F taskname,USERLIST D*
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST D*
ETBM0687 Participants:
ETBM0687 USER-ID ----------------------- C S P U E CHR SEQNO
ETBM0687 DOE                             N Y N N Y ASC 1
ETBM0687 DOE1                            N Y N N Y EBC 2
ETBM0687 DOE2                            N Y N N Y EBC 3
ETBM0687 DOE3                            N Y N N Y EBC 4
ETBM0582 Function completed
```

To display all participants with 4-character user ID, starting with uppercase "D" and with uppercase "E" as third character

- Enter command:

```
/F taskname,USERLIST D?E?
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST D?E?
ETBM0687 Participants:
ETBM0687 USER-ID ----------------------- C S P U E CHR SEQNO
ETBM0687 DOE1                            N Y N N Y EBC 2
ETBM0687 DOE2                            N Y N N Y EBC 3
ETBM0687 DOE3                            N Y N N Y EBC 4
ETBM0582 Function completed
```

Description of USERLIST Output Columns

Keyword	Description
USER-ID	User ID (32 bytes, case-sensitive). See USER-ID under *Broker ACI Fields*.
C	Client. Y Participant is a client, otherwise "N".
S	Server. Y Participant is a server, otherwise "N".
E	Big endian. Y Participant is on a big-endian machine. N Participant is on a little-endian machine.
CHR	Character set. ASC Participant is an ASCII user. EBC Participant is an EBCDIC user.
SEQNO	Sequence number of participant. Can be used for operator command CANCEL *parameter*.

USERS parameter

Operator command USERS displays selected user data of participant entries. The following parameters are supported:

Parameter	Description
none | *	Display all participants.
user_id	Display all participants with user ID *user_id*. Wildcard characters are supported.

Sample Commands

To display all participants

- Enter command:

```
/F taskname,USERS
```

Or:

```
/F taskname,USERS *
```

To display all participants with user ID "DOE"

- Enter command:

```
/F taskname,USERS DOE
```

This produces the following output. See *Description of USERS Output Columns* below.

```
ETBM0720 Operator typed in: USERS DOE
ETBM0687 Participants:
ETBM0687 USER-ID: DOE
ETBM0687 CLIENT: N SERVER:
ETBM0687 SEQNO: 6 BIG ENDIAN: Y CHARSET: ASCII PUID:
ETBM0687 202073756E6578322D2D3030303030324646462D2D3030303030303031
ETBM0687 TOKEN:
ETBM0582 Function completed
```

Description of USERS Output Columns

Keyword	Description
USER-ID	User ID (32 bytes, case-sensitive). See USER-ID under *Broker ACI Fields*.
CLIENT	Y Participant is a client, otherwise "N".
SERVER	Y Participant is a server, otherwise "N".
BIG ENDIAN	Y Participant is on a big-endian machine.

Keyword	Description
	N Participant is on a little-endian machine.
CHARSET	ASC Participant is an ASCII user. EBC Participant is an EBCDIC user.
PUID	Internal unique ID of participant. Hexadecimal 28-byte value in printable format.
TOKEN	Optionally identifies the participant. See TOKEN under *Broker ACI Fields*.

Security-specific Commands

DSECSTAT

Displays the number of successful and failed Security authentications and Security authorizations.

Sample Output

```
ETBM0720 Operator typed in: DSECSTAT
ETBM0579 Security Authentications - successful: 20 failed: 0
ETBM0579 Security Authorizations  - successful: 0 failed: 0
```

RESET userid

Resets the Security context for the specified user ID.

Sample Output

```
ETBM0720 Operator typed in: RESET EXXBATCH
ETBM0578 Reset ACEE for SAF-ID EXXBATCH : 20 instances found
```

SECURITY TRACE

Modifies the trace level for the EntireX Security (security-specific attribute `TRACE-LEVEL`). Broker-specific attribute `SECURITY=YES` must be set.

Example

To set a trace level 2 for EntireX Security

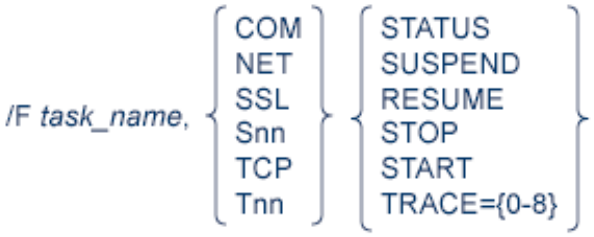
- Enter command:

```
/F taskname,SECURITY TRACE=2
```

See `TRACE-LEVEL` under *Security-specific Broker Attributes*.

Transport-specific Commands

Transport-specific commands are available for Adabas/Entire Net-Work communicators, SSL communicators and TCP communicators; the COM command can be used for all communicators. The following command syntax applies:



COM parameter

This command is executed by all configured transport communicators. The following parameters are supported:

Parameter	Description
STATUS	Displays the current status of the transport communicator.
SUSPEND	Used to suspend the transport communicator. The transport communicator is halted but will not shut down. User requests receive response code 148.
RESUME	Resume a suspended transport communicator. If the communicator was not suspended before, an error message will be displayed.
STOP	Stop an active or suspended transport communicator. The transport communicator will shut down. All transport-specific resources will be freed. User requests receive response code 148.
START	Start a transport communicator that was previously stopped. If the communicator was not stopped before, an error message will be displayed.
TRACE	Sets the trace level for the transport method. If the global trace level (see TRACE) is set with command /F *taskname*,TRACE=*n* this applies to *all* transport methods. This command will also override any existing transport-specific settings. If you subsequently enter command /F *taskname*, TCP TRACE=*n* only the trace level for TCP/IP transport is modified. **Note:** With commands TCP T*nn*, and SSL and S*nn*, the trace level is set for *all* TCP and SSL communicators respectively. Setting a trace level for a single TCP or SSL instance is not supported. For example: although it is possible to submit the command /F *taskname*,T01 TRACE=1 this command sets the trace level for all TCP communicators.

Sample Output

```
ETBM0720 Operator typed in: COM STATUS
ETBW0718 TCP Communicator 0 currently active
ETBW0718 TCP Communicator 1 currently active
ETBW0718 SSL Communicator 0 currently suspended
ETBW0718 NET Communicator 0 currently suspended
XCO0039I 00113 Total number of commands = 17
XCO0057I 00113 Operator entry active
ETBM0720 Operator typed in: COM SUSPEND
ETBM0721 TCP Communicator 0 suspended
ETBM0721 TCP Communicator 1 suspended
ETBM0721 SSL Communicator 0 suspended
ETBM0721 NET Communicator 0 suspended
```

NET parameter

This command is executed by X-COM, the Adabas/Entire Net-Work communicator. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: NET STATUS
ETBW0718 NET Communicator 0 currently active
XCO0039I 00113 Total number of commands = 17
XCO0057I 00113 Operator entry active
```

SSL parameter

This command is executed by all SSL communicators. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: SSL STATUS
ETBW0718 SSL Communicator 0 currently active
```

To manipulate a specific communicator instance (max. five instances can be started), use the command `S00`, `S01`, `S02`, `S03` or `S04` for the respective SSL instance.

TCP parameter

This command is executed by TCP communicators. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: TCP STATUS
ETBW0718 TCP Communicator 0 currently active
ETBW0718 TCP Communicator 1 currently active
```

```
ETBM0720 Operator typed in: TCP RESUME
ETBM0721 TCP Communicator 0 resumed
ETBM0721 TCP Communicator 1 resumed
```

To manipulate a specific communicator instance (max. five instances can be started), use the command `T00`, `T01`, `T02`, `T03` or `T04` for the respective TCP instance.

Sample Output

```
ETBM0720 Operator typed in: T00 STATUS
ETBW0718 TCP Communicator 0 currently active
```

```
ETBM0720 Operator typed in: T01 STATUS
ETBW0718 TCP Communicator 1 currently active
```

Sample Transport Commands

To display status of all transport communicators

- Enter command:

```
/F taskname,COM STATUS
```

To suspend first TCP communicator

- Enter command:

```
/F taskname,T00 SUSPEND
```

To stop all SSL transport communicators

- Enter command:

```
/F taskname,SSL STOP
```

XCOM-specific Commands

Note: All operator commands beginning with "X" belong to X-COM, the Adabas/Entire Net-Work communicator. The following commands operate only on the Adabas transport mechanism: `XCQES`, `XHALT`, `XPARM`, `XSTART`, `XSTAT` and `XUSER`. These commands have no effect on functions not related to the Adabas transport mechanism.

`XEND` and `XSTOP` function independently of the transport mechanism. (They stop the Broker's processing immediately, whereby existing calls to the EntireX Broker are not allowed to finish.)

XABS

Displays the total size, the number of bytes in use, the number of free bytes and the largest free windows in the Adabas attached buffer pool on the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XABS
XC00090I 10113 Attached buffer usage
XC00090I 10113   38912000 bytes total = 9500 NABS
XC00090I 10113          0 bytes used
XC00090I 10113          0 bytes used HWM
XC00090I 10113   38912000 bytes free
XC00090I 10113   38912000 bytes current largest free windows
XC00090I 10113   38912000 bytes minimum of all largest free windows
```

XCQES

Displays the current number, and the highest number, of Adabas command queue elements to the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XCQES
XC00030I 00113 Number of active CQEs = 0
XC00031I 00113 Highest number of active CQEs = 1
```

XEND

Alias of ETBEND.

XHALT

New calls to the EntireX Broker are temporarily rejected. Processing is resumed by issuing the XSTART operator command. XHALT is an alias for command NET SUSPEND.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XHALT
ETBM0721 NET Communicator 0 suspended
```

XPARM

Displays the values of Adabas SVC, database ID, number of CQEs, number of attached buffers, and the application name for the Adabas transport to the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XPARM
XC00032I 00113 Parameters for this session:
XC00033I 00113 SVC = 249
XC00034I 00113 NODE = 00113
XC00035I 00113 NCQE = 00100
XC00036I 00113 NABS = 10000
XC00037I 00113 User application = ETBNUC
```

XSTART

Processing of new calls to the EntireX Broker, interrupted with the `XHALT` command, is resumed. `XSTART` is an alias of command `NET RESUME`.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XSTART
ETBM0721 NET Communicator 0 resumed
```

XSTAT

Displays the EntireX Broker statistics as console messages.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

XSTOP

Alias of `ETBEND`.

XUSER

Displays the current number, as well as the highest number, of users actively issuing commands using the Adabas transport mechanism to the console.

Note: The number of users displayed with this operator command will not represent all of the Broker clients and servers but only the subset of users issuing commands using the Adabas transport mechanism. Command and Information Services provides comprehensive information about all Broker clients and servers.

Application Monitoring-specific Commands

This section covers the following topics:

APPMON=NO|YES

Use `APPMON=NO` to turn off the Application Monitoring feature in Broker. In addition to changing the current status, `APPLICATION-MONITORING=NO` is written to the Broker attribute file.

Use `APPMON=YES` to turn on the Application Monitoring feature in Broker. In addition to changing the current status, `APPLICATION-MONITORING=YES` is written to the Broker attribute file.

COLLECTOR=host:port

Use `COLLECTOR=`*`host:port`* to set the collector broker ID in Broker. `COLLECTOR-BROKER-ID=`*`value`* is written to the Broker attribute file. If the `APPLICATION-MONITORING` section is not already defined in the attribute file, the section is added, that is, a line containing `DEFAULTS = APPLICATION-MONITORING` followed by attribute `COLLECTOR-BROKER-ID=`*`value`*.

DISPLAY APPMON

Use `DISPLAY APPMON` to show the current settings of Application Monitoring. Sample output:

```
ETBM0720 Operator typed in: DISPLAY APPMON
ETBM0793 APPLICATION-MONITORING=YES
ETBM0793 COLLECTOR-BROKER-ID=SUSEXX04C:3930
```

10 Tracing EntireX Broker

See also *Tracing for Broker Stubs*.

Switching on Tracing

To switch on tracing

- Set the attribute TRACE-LEVEL in the broker attribute file
  - for minimal trace output to "1"
  - for detailed trace output to "2"
  - for full trace output to "3"

  Example:

```
TRACE-LEVEL=2
```

See also *EntireX Broker Return Codes*.

Switching off Tracing

To switch off tracing

- Set the attribute TRACE-LEVEL in the broker attribute file to 0:

```
TRACE-LEVEL=0
```

  Or:

  Omit the TRACE-LEVEL attribute.

Deferred Tracing

It is not always convenient to run with TRACE-LEVEL defined, especially when higher trace levels are involved. Deferred tracing is triggered when a specific condition occurs, such as an ACI response code or a broker subtask abend. Such conditions cause the contents of the trace buffer to be written, showing trace information leading up the specified event. If the specified event does not occur, the Broker trace will contain only startup and shutdown information (equivalent to TRACE-LEVEL=0).

Operating the trace in this mode requires the following additional attributes in the broker section of the attribute file. Values for TRBUFNUM and TRAP-ERROR are only examples.

Attribute	Value	Description
TRBUFNUM	3	Specifies the deferred trace buffer size = 3 * 64 K.
TRMODE	WRAP	Indicates trace is not written until an event occurs.
TRAP-ERROR	322	Assigns the event ACI response code 00780322 "PSI: UPDATE failed".

Dynamically Switching On or Off the EntireX Broker Trace

The following methods are available to switch on or off the EntireX Broker trace dynamically. You do not need to restart the broker for the changes to take effect.

- ETBCMD
  Run command utility ETBCMD with option -c TRACE-ON or - c TRACE-OFF. See ETBCMD.
- **Operator Command**
  Issue an operator command. See TRACE.

  See also *Deferred Tracing*.

11 Broker Shutdown Statistics

Shutdown Statistics Output

After a successful Broker execution, shutdown statistics and related information are produced. This output is written in the following sequence:

1. The diagnostic message ETBD0444 is written into the Broker trace log.
2. The output - i.e. statistics, internals and user-specified parameters - is written into the end of the Broker trace log file at shutdown.

Table of Shutdown Statistics

See *Legend* below for explanation of output type.

Output Type	Display Field	Description
U	Broker ID	Identifies the Broker kernel to which the attribute file applies. See `BROKER-ID`.
I	Version	The version of the Broker kernel currently running.
I	Generated platform family	The platform family for which this Broker kernel was built.
I	Runtime platform	The platform on which this Broker kernel is currently running.
I	Start time	The date and time when this Broker kernel started.
S	Restart count	The restart count indicates how many times the Broker kernel has been started with the persistent store. Therefore, after a cold start `(PSTORE=COLD)`, the restart count will be 1. Then, after subsequent hot starts `(PSTORE=HOT)`, the restart count will be 2 or greater.
U	Trace level	The value for the trace setting for this Broker kernel. See `TRACE-LEVEL`.
U	Worker tasks	The number of worker tasks for this Broker kernel. See `NUM-WORKER`.
U	MAX-MEMORY	The value of `MAX-MEMORY` or 0 if not defined. See `MAX-MEMORY`.
S	Memory allocated	Size of the allocated memory, in bytes.
S	Memory allocated HWM	Highest size of allocated memory in bytes since Broker started.
U	NUM-SERVICE	Value of `NUM-SERVICE` or 0 if not defined. See `NUM-SERVICE`.
S	Services active	The number of services currently active for this Broker kernel.
U	NUM-CLIENT	Value of `NUM-CLIENT` or 0 if not defined. See `NUM-CLIENT`.
S	Clients active	The number of clients currently active for this Broker kernel.
S	Clients active HWM	The high watermark for the number of clients active for this Broker kernel.

Output Type	Display Field	Description
U	NUM-SERVER	Value of NUM-SERVER or 0 if not defined. See NUM-SERVER.
S	Servers active	The number of servers currently active for this Broker kernel.
S	Servers active HWM	The high watermark for the number of servers active for this Broker kernel.
U	NUM-CONVERSATION	Value of NUM-CONVERSATION or 0 if not defined. See NUM-CONVERSATION.
S	Conversations active	The number of conversations currently active for this Broker kernel.
S	Conversations active HWM	The high watermark for the number of conversations active for this Broker kernel.
U	NUM-LONG-BUFFER	Value of NUM-LONG-BUFFER or 0 if not defined. See NUM-LONG-BUFFER.
S	Long buffers active	The number of long message buffers currently in use for this Broker kernel.
S	Long buffers active HWM	The high watermark for the number of long message buffers used for this Broker kernel.
U	NUM-SHORT-BUFFER	Value of NUM-SHORT-BUFFER or 0 if not defined. See NUM-SHORT-BUFFER.
S	Short buffers active	The number of short message buffers currently in use for this Broker kernel.
S	Short buffers active HWM	The high watermark for the number of short message buffers used for this Broker kernel.
U	Persistent store type	The type of persistent store used by this Broker kernel. See PSTORE-TYPE.
U	UOW persistence	Indicates whether units of work are persistent or not in this Broker kernel. See STORE.
U	Persistent store startup	Indicates the status of the persistent store at Broker startup. See PSTORE.
U	Persistent status lifetime	The multiplier to compute the lifetime of the persistent status. See UWSTATP.
U	Deferred UOWs allowed	Indicates whether or not deferred units of work are allowed. See DEFERRED.
U	Maximum allowed UOWs	The maximum number of units of work that can be active concurrently for this Broker kernel. See MAX-UOWS.
U	Maximum messages per UOW	The maximum number of messages allowed in a unit of work. See MAX-MESSAGES-IN-UOW.
U	UOW lifetime in seconds	Indicates the default lifetime for a unit of work. See UOW-DATA-LIFETIME.
U	Maximum message length	Indicates the maximum message size that can be sent. See MAX-UOW-MESSAGE-LENGTH.

Output Type	Display Field	Description
U	New UOW messages allowed	Indicates whether or not new units of work are allowed in this Broker kernel. See NEW-UOW-MESSAGES.
S	UOWs active	The number of units of work currently active in this Broker kernel.
S	Current UOW	The number of the last unit of work in this Broker kernel.
U	Accounting	Indicates the status of accounting records in this Broker kernel. See ACCOUNTING.
U	SSL port *	If applicable, the SSL port number on which this Broker kernel will listen for connection requests. See SSL-specific attribute PORT.
U	TCP port *	If applicable, the TCP port number on which this Broker kernel will listen for connection requests. See TCP-specific attribute PORT.
I	Number of function calls	Marks the beginning of the section of summary statistics for all the function calls.
S	DEREGISTER	The number of Broker DEREGISTER function calls since startup.
S	EOC	The number of Broker EOC function calls since startup.
S	KERNELVERS	The number of Broker KERNELVERS function calls since startup.
S	LOGOFF	The number of Broker LOGOFF function calls since startup.
S	LOGON	The number of Broker LOGON function calls since startup.
S	RECEIVE	The number of Broker RECEIVE function calls since startup.
S	REGISTER	The number of Broker REGISTER function calls since startup.
S	SEND	The number of Broker SEND function calls since startup.
S	SYNCPOINT	The number of Broker SYNCPOINT function calls since startup.
S	UNDO	The number of Broker UNDO function calls since startup.
S	REPLY_ERROR	The number of Broker REPLY_ERROR function calls since startup.
I	Worker task statistics	Marks the beginning of the section of summary statistics for all the worker tasks.
I	Worker number	The identifier of the worker task.
I	Status	The status of the worker task at shutdown.
S	# of calls	The number of Broker calls handled by the worker task since startup.
S	Idle time in seconds	The number of seconds the worker task has been idle since startup.

* Does not apply to z/OS.

Legend

Output Type	Description	Value	Origin of Value
I	Internal Information	Static	Determined by Software AG EntireX.
S	Shutdown Statistic	Variable	Determined by Broker activity during execution.
U	User-Specified Parameter	Variable	Specified by Broker administrator before or, if allowable, during execution.

12 Command Logging in EntireX

Command logging is a feature to assist in debugging Broker ACI applications. A command in this context represents one user request sent to the Broker and the related response of Broker.

Command logging is a feature that writes the user requests and responses to file in a way it is already known with Broker trace and `TRACE-LEVEL=1`. But command logging works completely independent from trace, and data is written to a file only if defined command trace filters detect a match.

Broker stub applications send commands or requests to the Broker kernel, and the Broker kernel returns a response to the requesting application. Developers who need to resolve problems in an application need access to those request and response strings inside the Broker kernel. That's where command logging comes in. With command logging, request and response strings from or to an application are written to a file that is separate from the Broker trace file. Command logging works based on defined filters. Nothing is logged if there are no filters. If filters are defined and if there is a match, this user request is logged.

Note: All applied filters are lost after Broker restart and have to be applied again.

Introduction to Command Logging

This section provides an introduction to command logging in EntireX and offers examples of how command logging is implemented. It covers the following topics:

- Overview
- Command Log Files
- Defining Filters
- Programmatically Turning on Command Logging

Overview

Command logging is similar to a Broker trace that is generated when the Broker attribute `TRACE-LEVEL` is set to 1. Broker trace and command logging are independent of each other, and therefore the configuration of command logging is separate from Broker tracing.

The following Broker attributes are involved in command logging:

Attribute	Description
`CMDLOG`	Set this to "N" if command logging is not needed.
`CMDLOG-FILE-SIZE`	A numeric value indicating the maximum size of command log file in KB.
`NUM-CMDLOG-FILTER`	The maximum number of filters that can be set.

In addition to `CMDLOG=YES`, the Broker needs the assignment of the dual command logging files during startup. If these assignments are missing, Broker will set `CMDLOG=NO`. See also *Broker Attributes*.

Command Log Files

The Broker keeps a record of commands (request and response strings) in a command log file.

At Broker startup, you will need to supply two command log file names and paths. Only one file is open at a time, however, and the Broker writes commands (requests and responses) to this file.

Under z/OS, the file requirements are two equally sized, physical sequential files defined with a record length of 121 bytes, i.e. `DCB=(LRECL=121,RECFM=FB,BLKSIZE=nnnn)`. We recommend you allocate files with a single (primary) extent only. For example `SPACE=(CYL,(30,0))`. The minimum file size is approximately 3 cylinders of 3390 device. Alternatively, the dual command log files can be allowed in USS HFS file system.

- If the value of `CMDLOG-FILE-SIZE` is lower than the capacity of the CMDLOG data set, `CMDLOG-FILE-SIZE` is used.
- If the value of `CMDLOG-FILE-SIZE` is greater than the capacity of the CMDLOG data set, `CMDLOG-FILE-SIZE` is adjusted to the capacity of the data set.
- If no `CMDLOG-FILE-SIZE` was defined, it is set to the capacity of the CMDLOG data set.

When the size of the active command log file reaches the KB limit set by `CMDLOG-FILE-SIZE`, the file is closed and the second file is opened and becomes active. When the second file also reaches the KB limit set by `CMDLOG-FILE-SIZE`, the first file is opened and second file is closed. Existing log data in a newly opened file will be lost.

Defining Filters

In command logging, a filter is used to store and identify a class, server, or service, as well as a user ID.

Use the command-line tool `etbcmd` to define a filter. During processing, the Broker evaluates the class, server, service, and user ID associated with each incoming request and compares them with the same parameters specified in the filters. If there is a match, the request string and response string of the request is printed out to the command log file.

Programmatically Turning on Command Logging

Applications using ACI version 9 or above have access to the new field `LOG-COMMAND` in the ACI control block.

If this field is set, the accompanying request and the Broker's response to this request is logged to the command log file.

Note: Programmatic command logging ignores any filters set in the kernel.

Command Log Filtering using Command-line Interface ETBCMD

The examples assume that Broker has been started with the attribute `CMDLOG=Y`.

Setting Filters

Filters need to be set before running the stub applications whose commands are to be logged. Filter for class, server, service may contain fully qualified names (`ACLASS/ASERVER/ASERVICE`) or asterisk for any (e.g. `ACLASS/*/ASERVICE`). Partially qualified filter names (`ACLA*/ASERVER/ASERV*`) are not supported.

Command	Description

`//ETBCMD  EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP ↩`
`-cSET-CMDLOG-FILTER -xuser ',`
`//       '-dBROKER ↩`
`-nACLASS/ASERVER/ASERVICE')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE`. All ACI calls issued by *all* users to this service will be logged.

`//ETBCMD  EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP ↩`
`-cSET-CMDLOG-FILTER -xuser ',`
`//      '-dBROKER -nACLASS/ASERVER/ASERVICE ↩`
`-Usaguser1')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE` and user ID `saguser1`. All ACI calls to this service *as well as* those issued by `saguser1` will be logged.

Note: If more than one service is set as a filter, all ACI calls sent to any of these services will be logged. Identical filters cannot be set. Attempts to set a second filter that matches an existing filter will be rejected. Similarly, the maximum number of filters that can be added is defined in `NUM-CMDLOG-FILTER`. If the maximum number of filters is already being used, delete an existing filter to make room for a new filter.

Deleting Filters

The following provides an example of how to delete an existing filter on a service.

To delete a filter

- Enter the following command.

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cCLEAR-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE')
```

  If the filter does not exist, the command will return an error.

Disabling and Enabling a Filter

Filters can be set and still be disabled (made inactive).

To disable a filter

- Enter the following command.

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

  Note: A disabled filter will not bring down the count of filters in use.

To enable a filter

- Enter the following command to enable the disabled filter.

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

ACI-driven Command Logging

EntireX components that communicate with Broker can trigger command logging by setting the field `LOG-COMMAND` in the ACI control block.

When handling ACI functions with command log turned on, Broker will not evaluate any filters. Application developers must remember to reset the `LOG-COMMAND` field if subsequent requests are not required to be logged.

Dual Command Log Files

Broker's use of two command log files prevents any one command log file from becoming too large.

When starting a Broker with command log support, you must therefore specify two data sets and DD names - one for each of the two command log files. The sample started task EXBSTART delivered with the EXX107.JOBS data set uses DDCLOGR1 and DDCLOGR2 as default command log file names.

At startup, Broker initializes both files and keeps one of them open. Command log statements are printed to the open file until the size of this file reaches the value specified in the Broker attribute `CMDLOG-FILE-SIZE`. This value must be specified in KB.

When the size of the open file exceeds the value specified in the Broker attribute `CMDLOG-FILE-SIZE`, Broker closes this file and opens the other, dormant file. Because the Broker closes a log file only when unable to print out a complete log line, the size of a *full* file may be smaller than `CMDLOG-FILE-SIZE`.

To switch log files on demand, using `ETBCMD`

- An open command log file can be forcibly closed even before the size limit is reached. Enter the following command.

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cSWITCH-CMDLOG -xuser ',
//       '-dBROKER')
```

The command above will close the currently open file and open the one that has been dormant.

13 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	Type of SMF record.
Record Write Time	1	I4I4 timestamp	SMF timestamp in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)).
SMF system ID	1	4-byte string	ID of the SMF system.
SMF subsystem ID	1	4-byte string	ID of the SMF subsystem.
EntireX Broker ID	1	A32	Broker ID from attribute file.
EntireX Version	1	A8	Version information, *v.r.s.p* where *v* =version *r* =release *s* =service pack *p* =patch level for example 10.7.0.00.
Platform of Operation	1	A8	Platform where EntireX is running.
EntireX Start Time	1	I4I4 timestamp	The time EntireX was initialized in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)).
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 = IBM® MQ 5 = SSL
Client Requests Made	1	I4	Number of Requests made by client.
Client Sent Bytes	1	I4	Number of bytes sent by client.
Client Received Bytes	1	I4	Number of bytes received by client.
Client Sent Messages	1	I4	Number of messages sent by client.
Client Received Messages	1	I4	Number of messages received by client.
Client Sent UOWs	1	I4	Number of UOWs sent by client.
Client UOWs Received	1	I4	Number of UOWs received by client.
Client Completion Code	1	I4	Completion code client received when conversation ended.
Server User ID	1	A32	USER-ID ACI field from the server in the conversation.
Server Token	1	A32	TOKEN field from the ACI from the server.
Server Physical ID	1	A56	The physical user ID of the server, set by EntireX.
Server Communication Type	1	I1	Communication used by Server: 1 = Entire Net-Work 2 = TCP/IP 3 = APPC 4 = IBM® MQ 5 = SSL
Server Requests Made	1	I4	Number of requests made by server.
Server Sent Bytes	1	I4	Number of bytes sent by server.
Server Received Bytes	1	I4	Number of bytes received by server.
Server Sent Messages	1	I4	Number of messages sent by server.
Server Received Messages	1	I4	Number of messages received by server.
Server Sent UOWs	1	I4	Number of UOWs sent by server.
Server Received UOWs	1	I4	Number of UOWs received by server.
Server Completion Code	1	I4	Completion code server received when conversation ended.

Field Name	Accounting Version	Type of Field	Description
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	I4I4 timestamp	The time the conversation began in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)).
Conversation End Time	1	I4I4 timestamp	The time the conversation was cleaned up in format I4I4 (time in hundredths of seconds followed by date in format X'0CYYDDDF' (packed decimal number)).
Conversation CPU Time	1	I4	Number of microseconds of CPU time used by the conversation
Client Security Identity	2	A32	Actual identity of client derived from authenticated user ID.
Client Application Node	2	A32	Node name of machine where client application executes.
Client Application Type	2	A8	Stub type used by client application.
Client Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field APPLICATION-NAME.
Client Credentials Type	2	I1	Mechanism by which authentication is performed for client.
Server Security Identity	2	A32	Actual identity of server derived from authenticated user ID.
Server Application Node	2	A32	Node name of machine where server application executes.
Server Application Type	2	A8	Stub type used by server application.
Server Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field APPLICATION-NAME.
Server Credentials Type	2	I1	Mechanism by which authentication is performed for server.
Client RPC Library	3	A128	RPC library referenced by client when sending the only/first request message of the conversation.

Field Name	Accounting Version	Type of Field	Description
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 z/OS

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

Attribute File

ACCOUNTING={NO|128-255}

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

Default value: NO

Retrieving Accounting Records

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

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

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

Accounting Record Layouts

EntireX provides three mappings for its accounting records in the following members, all located in the EXX107.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.