Administering Broker Stubs

This chapter covers the following topics:

- Available Stubs
- Transport Methods for Broker Stubs
- Tracing for Broker Stubs
- Application Stublog File
- UNIX Commands to Set the Environment Variables
- Support of Clustering in a High Availability Scenario

Available Stubs

The following table lists available stubs and gives an overview of available features and supported transport methods.

Stub	Language	Transport Methods	Compression	More Information
Jaci	Java	TCP /SSL	Yes	See *Java ACI*.
broker.s[o|l]	C	TCP / SSL	Yes	See below.

Transport Methods for Broker Stubs

The Broker stub can use TCP/IP and SSL. In this section, "SSL" refers to both Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

- Using TCP/IP as Transport Method for the Broker Stub
- Using SSL or TLS as Transport Method for the Broker Stub
- Setting the Timeout for the Transport Method
- Limiting the TCP/IP Connection Lifetime
- Modifying the Hosts and Services Tables

Using TCP/IP as Transport Method for the Broker Stub

To use TCP/IP

1. Optional: set the timeout, see *Setting the Timeout for the Transport Method*.
2. The Broker stub requires the IP address and the TCP port number (if the Broker's default TCP port number 1971 cannot be used) for each BROKER-ID. Either add an entry in the Domain Name System (DNS) or modify your local hosts and services tables. See *Modifying the Hosts and Services*

Tables.

You can check whether the Broker has already been added to your DNS with the command:

```
ping <broker-id>
```

for example: ping ETB001. If a message such as "...is alive" or "Reply from ..." is displayed (the text displayed varies depending on your ping implementation), the name is known to your DNS and the host where the Broker is running is reachable. However, this does not necessarily mean that the Broker is active.

Using SSL or TLS as Transport Method for the Broker Stub

Note:
The `SETSSLPARMS` function must contain the subparameter `VERIFY_SERVER=N`, unless the common name of the server certificate matches the Broker name. Otherwise, the connection will be refused.

Example:

```
TRUST_STORE=/opt/softwareag/EntireX/etc/ExxCACert.pem&VERIFY_SERVER=N
```

To use Secure Sockets Layer

1. To operate with Secure Sockets Layer, certificates need to be provided and maintained. Software AG provides default certificates, but we strongly recommend that you create your own, for example, with the OpenSSL toolkit. The certificates must be installed locally with the EntireX Broker Stub.

2. Set the value SSL as part of the Broker ID (see the field `BROKER-ID` in the ACI control block, see also *Using the Broker ID in Applications*) and set the SSL parameters (see *Setting SSL or TLS Parameters*). Example: localhost:1958:SSL.

   The SSL parameters can be specified with the `FCT_SETSSLPARMS` call type for ACI programs, or they can be appended with a "?" to the broker ID (Java stub).

3. The Broker stub requires the IP address and the SSL port number for each `BROKER-ID`. Either add an entry to the Domain Name System (DNS) or modify your local hosts and services tables. See *Modifying the Hosts and Services Tables*.

   The default port number is 1958.

   You can check whether the Broker has already been added to your DNS with the following command:

   ```
   ping <broker-id>
   ```

   for example: ping ETB001. If a message such as "...is alive" or "Reply from ..." is displayed (the text displayed varies depending on your ping implementation), the name is known to your DNS and the host where the Broker is running is reachable. However, this does not necessarily mean that the Broker is active.

Setting SSL or TLS Parameters

Enter the SSL parameters as follows: <*keyword*>=<*value*>. Parameters are separated by "&".

Example code:

```
/opt/softwareag/EntireX/examples/ACI/conversational/C/convSvr -blocalhost:1958:SSL -cACLASS -sASERVER -vASERVICE -x
"VERIFY_SERVER=N&TRUST_STORE=/opt/softwareag/EntireX/etc/ExxCACert.pem"
```

Warning:
If stub tracing level is > 1, unencrypted contents of the send/receive buffers are exposed in the trace.

For information on the parameters see *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows.

Setting the Timeout for the Transport Method

The timeout settings of the transport layers are independent of the broker's timeout settings, which are set by the application in the `WAIT` field of the broker ACI control block.

If the transport layer is interrupted, communication between the Broker and the stub (i.e. client or server application) is interrupted as well. To prevent a client from waiting for a Broker reply indefinitely, set a timeout value for the transport method. The actual timeout for the procedure is then the Broker timeout (which is set by the application in the `WAIT` of the broker ACI control block) plus this value.

To set a transport timeout value

- Set the environment variable `ETB_TIMEOUT`:

Transport Timeout Value	Description
0	Infinite wait for the application.
n	Transport method waits additional time in seconds. A negative value is treated as `ETB_TIMEOUT=0` (infinite wait).
No environment variable defined	Transport method waits additional 20 seconds.

See also *UNIX Commands to Set the Environment Variables*.

Limiting the TCP/IP Connection Lifetime

With transport methods TCP/IP and SSL, 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`.

Transport Non-activity Value	Description
0	Infinite lifetime until application is stopped.
n (seconds)	Transport connections with non-activity time greater than *n* will be closed.
Nothing set	Transport connections with non-activity time greater than 300s (default) will be closed.

Modifying the Hosts and Services Tables

The Hosts and Services tables are plain text files in directory */etc*.

To add an entry to the hosts table

- Add a line similar to the following to the local hosts file:

```
100.100.1.1 ETB226 # ETB test host name
```

To add an entry to the services table

- Add lines similar to the following to the local services file:

```
ETB226 18492/tcp # ETB test host name
ETB411 21234/tcp # ETB production host name
```

Tracing for Broker Stubs

The broker stubs provide an option for writing trace files.

To switch on tracing for the broker stub

- Before starting the client application, set the environment variable `ETB_STUBLOG`:

Trace Value	Trace Level	Description
0	`NONE`	No tracing.
1	`STANDARD`	Traces initialization, errors, and all ACI request/reply strings.
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 transport information.
3	`SUPPORT`	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Example:

```
ETB_STUBLOG=2
```

If the trace level is greater than 1, unencrypted contents of the send/receive buffers may be exposed in the trace.

The trace file is created in the current directory. The name is *pid.etb* where *pid* is the process ID. If you want to write the trace file to a different location, set environment variable ETB_STUBLOGPATH to the desired path.

See also *UNIX Commands to Set the Environment Variables*.

Remember to switch off tracing to prevent trace files from filling up your disk.

To switch off tracing for the broker stub

- Set the environment variable ETB_STUBLOG to NONE or delete it.

Application Stublog File

Logging works for both TCP and SSL. Tracing is controlled by the environment variable ETB_STUBLOG.

csh or tcsh users use:

```
setenv ETB_STUBLOG tracelevel
```

bsh, ksh or bash users use:

```
ETB_STUBLOG=tracelevel; export ETB_STUBLOG
```

Possible values for *tracelevel*:

Trace Value	Trace Level	Description
0	NONE	No tracing.
1	STANDARD	Traces initialization, errors, and all ACI request/reply strings.
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 transport information.
3	SUPPORT	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

If you start your application with this environment variable set, a log file is created in the directory where your application is started. The name of the log file is *pid.etb*

csh or tcsh users use:

```
unsetenv ETB_STUBLOG
```

bsh, ksh or bash users use:

```
unset ETB_STUBLOG
```

UNIX Commands to Set the Environment Variables

Example of ETB_TRANSPORT:

Shell	set the environment variable:	delete the environment variable:
C Shell	`setenv ETB_TRANSPORT value`	`unsetenv ETB_TRANSPORT`
Bourne or Korn Shell	`ETB_TRANSPORT=value` `export ETB_TRANSPORT`	`unset ETB_TRANSPORT`

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:

- Introduction
- Exceptions
- Default

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.

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.