Administration of Broker Stubs under z/VM

This chapter covers the following topics:

- Available Stubs
- BKIMBCMS
- Specifying the Broker Stub in Natural Environments
- Transport Methods for Broker Stubs
- Tracing the Broker Stub under z/VM
- Configuring Adabas Communication in your Environment
- Setting Environment Variables under z/VM

Available Stubs

This table lists all Broker stubs available under the z/VM operating system which are to be used with the programming languages Assembler | C | COBOL | Natural | PL/I.

If your Broker kernel is running on another mainframe operating system, such as z/OS or z/VSE, you can use transport method NET or transport method TCP to communicate with the Broker kernel from z/VM.

If your Broker kernel is running under UNIX or Windows, you can use transport method TCP to communicate with the Broker kernel from z/VM.

Note:
z/VM stub does not currently support send/receive buffers greater than 32 KB when using transport method NET.

Environment	Transport			Transport Timeout	Trace	Compression	Stub Module
	NET	TCP	SSL				
• z/VM	Yes	Yes	No	Yes	Yes	No	*BKIMBCMS* *

* BKIMBCMS does not support multi-threading or multitasking applications.

BKIMBCMS

Prerequisites and Installation Notes

This stub can be used with conventional as well as IBM Language Environment applications. It does not support multitasking applications.

Linkage

Link your application to the TEXT member BKIMBCMS from the EntireX distribution package. BKIMBCMS has the entry point BROKER.

Linkage using the IBM Binder for LE Programs

```
CMOD MYPROG BKIMBCMS
```

Linkage using IBM GENMOD for non-LE Programs

```
LOAD MYPROG
INCLUDE BKIMBCMS
GENMOD MYPROG
```

Specifying the Broker Stub in Natural Environments

Natural running under z/VM uses the Broker stub BKIMBCMS. The TEXT module BKIMBCMS can be linked with the Natural nucleus or loaded dynamically. However, you must not link BKIMBCMS with the Natural nucleus - for example, using NATBLDS - if executing in a shared segment. To load the Broker stub dynamically, use the following Natural parameter:

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

Transport Methods for Broker Stubs

Transport Method Values

The following table describes the possible values for the transport methods:

Transport Value	Description / Tips
NET	Use Adabas Cross-Memory Services as the transport method. 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/WAL V811 allows more than 32 KB of data to be communicated. Otherwise the following maximums are allowed: **ACI Version** — **Max Send/Receive length** 1 — 32167 2, 3 — 31647 4 - 8 — 31643 9 or above — 31123 **Note:** If Adabas version 8 is *not* used, these same limits still apply under z/OS.
TCP	Use TCP/IP as the transport method.
SSL	Use Secure Sockets Layer (SSL) as the transport method (not yet available).

Default Transport Methods

The default transport method for stub BKIMBCMS is TCP.

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.

Setting the Timeout for the Transport Method

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 settings for transport layers are independent of the Broker's.

The timeout value for the transport method 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.

The following table describes the possible values for the transport timeout.

Transport Timeout Value	Description
0	Infinite wait for the application.
n	The transport method waits this additional time in seconds. Negative values will be treated as `TIMEOUT=0` (infinite wait for the application).
nothing set	Transport method waits additional 20 seconds.

The actual timeout for transport layer = Broker timeout (`WAIT` field) + timeout value for transport method.

Tracing the Broker Stub under z/VM

Tracing is supported by the z/VM Broker stub and can be activated by setting the environment variable ETB_STUBLOG.

Allocating the Trace File

The stub trace file is allocated as follows:

```
FILEDEF ETBLOG DISK <file-name> <file-type>  <file-mode>
```

Trace Values

The following table describes the possible values for the Broker stub trace:

Trace Level	Description	
0	NONE	No tracing. Switch tracing off.
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 - the Broker ACI control block, for example - as well as information from the transports.
3	SUPPORT	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Configuring Adabas Communication in your Environment

To use the Adabas/Entire Net-Work transport to communicate between an application with a Broker on another mainframe, link the ADAENTRY module against the Adabas link module ADALNK. Ensure that your Adabas TEXT library is available before running the link.

```
LOAD ETBADUSR (RLDSAVE RESET ETBADUSR)

GENMOD ADAENTRY (AMODE 31 RMODE ANY)
```

Setting Environment Variables under z/VM

Language Environment programs can specify the environment variable - for example `ETB_STUBLOG` - in the command line:

```
MYPROG ENVAR(ETB_STUBLOG=1)/-bMYBROKER
```

Non-Language Environment applications require setting z/VM environments before executing the application:

```
GLOBALV SELECT CENV SETLP ETB_STUBLOG 1
```