z/OS

UNIX System Services Connection Scaling Reference for iBaanERP Solutions

SA22-7809-02

z/OS

UNIX System Services Connection Scaling Reference for iBaanERP Solutions

SA22-7809-02

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 83.

Third Edition, September 2002

This edition applies to Version 1 Release 4 of z/OS™ (5694-A01), to Version 1 Release 4 of z/OS.e™ (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM® representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1998, 2002. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures

© Copyright IBM Corp. 1998, 2002

Tables

© Copyright IBM Corp. 1998, 2002

About this document

This document describes the installation, programming interfaces, and problem determination tools available for the Connection Manager and Process Manager components of z/OS UNIX® System Services Application Services. This document is written for system programmers responsible for program installation and application programmers who want to create applications which use z/OS UNIX System Services (z/OS UNIX) and z/OS.e.

Organization

This document has the following parts:

- Chapter 1, "Introduction" on page 1 describes Connection Manager and Process Manager at a conceptual level.
- Chapter 2, "Post-installation, configuration, and activation" on page 3 describes the installation, configuration, and activation required for Connection Manager and Process Manager.
- Chapter 3, "Application program interface" on page 25 describes the Connection Manager and Process Manager programming interfaces.
- Chapter 4, "Problem determination and messages" on page 37 describes the Connection Manager and Process Manager problem determination tools and messages.

Where to find more information

Where necessary, this document references information in other documents about the elements and features of z/OS. For complete titles and order numbers for all z/OS documents, see *z/OS Information Roadmap*.

Direct your request for copies of any IBM publication to your IBM representative or to the IBM branch office serving your locality.

There is also a toll-free customer support number (1-800-879-2755) available Monday through Friday from 6:30 a.m. through 5:00 p.m. Mountain Time. You can use this number to:

- Order or inquire about IBM publications
- Resolve any software manufacturing or delivery concerns
- Activate the program reorder form to provide faster and more convenient ordering of software updates

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

1. z/OS.e customers received a Memo to Licensees, (GI10-0684) that includes this key code.

© Copyright IBM Corp. 1998, 2002

To obtain your IBM Resource Link user ID and password, log on to:

```
http://www.ibm.com/servers/resourcelink
```

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most of the z/OS, z/VM, and VSE messages you encounter, as well as system abends and some codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

```
http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/
```

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt on the Internet to find a message explanation, go to the LookAt Web site and simply enter the message identifier (for example, `$HASP701` or `$HASP*`). You can select a specific release to narrow your search.

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/.
2. Click **News**.
3. Scroll to **Download LookAt Code for TSO/E and z/VM.**
4. Click the ftp link, which will take you to a list of operating systems. Click the appropriate operating system. Then click the appropriate release.
5. Open the **lookat.me** file and follow its detailed instructions.

After you have LookAt installed, you can access a message explanation from a TSO/E command line by entering: **lookat** *message-id*. LookAt will display the message explanation for the message requested.

Note: Some messages have information in more than one document. For example, IEC192I can be found in *z/OS MVS System Messages, Vol 7 (IEB-IEE)* and also in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt displays a list of documents in which the message appears. You can then click the message identifier under each document title to view information about the message.

Softcopy publications

The z/OS UNIX library is available on the *z/OS Collection Kit* , SK2T-6700. This softcopy collection contains a set of z/OS and related unlicensed product documents. The CD-ROM collection includes the IBM Library Reader™, a program that enables customers to read the softcopy documents.

Softcopy z/OS publications are also available for web-browsing and PDF versions of the z/OS publications for viewing or printing using Adobe Acrobat Reader at this URL:

`http://www.ibm.com/servers/eserver/zseries/zos/`

Select “Library”.

Documentation conventions

This manual uses the following conventions:

Commands

Boldface characters indicate items that you type, such as commands and options. For example: **`C:\A INSTALL`**

Delimiter bar (|)

In syntax examples, a delimiter bar separating two command options indicates that you can choose one of the options.

For example:

-S | -R

Do **not** type the bar.

Ellipses

Ellipses (...) in syntax examples indicate that parameters, options, or settings can be repeated.

Emphasis

Italic type indicates emphasized text. For example:

Remember to load the driver *before* you install the application.

Key names

Angle brackets surround the name of a key. For example, <Enter> corresponds to the Enter key on your keyboard.

Options

In syntax examples, braces indicate that you are required to choose one of the enclosed options. For example, the following notation means that you must include a 0 or a 1 in the command:

{0, 1}

Square brackets

In syntax examples, boldface type enclosed in square brackets indicates command options that you can type as needed. For example:

FTP [**-D**] [**-F**]

System response

Monospace type shows system-generated responses that appear on your workstation screen. For example:

```
Program ended.
```

UNIX commands

UNIX commands are shown in boldface letters. For example, **vi**. Because UNIX is case-sensitive, these commands are usually lowercase. Type UNIX commands exactly as shown.

UNIX filenames, directory names, and pathnames

UNIX filenames, directory names, and pathnames are shown in italics. For example, */etc/hosts*

Because UNIX is case-sensitive, these names usually are in lowercase letters. Type UNIX filenames exactly as shown.

Variables

Italic type indicates variables—descriptive item names, such as command parameters—that you replace with appropriate values.

For example, in the command:

FTP -F *remote_host*

you type the name of a computer on your network in place of *remote_host*.

Summary of changes

Summary of changes for SA22-7809-02 z/OS Version 1 Release 4

This document contains information previously presented in *z/OS UNIX System Services Connection Scaling Reference*, SA22-7809-01, which supports z/OS Version 1 Release 3.

Changed information

- Replaced all instances of SEZALINK with SEZALOAD
- Replaced all instances of AEZAMOD1 with AEZAMODS

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes for SA22-7809-01 z/OS Version 1 Release 3

This document contains information previously presented in *z/OS UNIX System Services Connection Scaling Reference*, SA22-7809-00, which supports z/OS Version 1 Release 2.

New information

- Information on using JCL to start Process Manager has been added to the document.
- An appendix with z/OS product accessibility information has been added.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

© Copyright IBM Corp. 1998, 2002

Chapter 1. Introduction

z/OS UNIX System Services Connection Scaling provides two components:
- Connection Manager
- Process Manager

Connection Manager

Connection Manager provides a set of ODBC-like interfaces to DB2® for z/OS. Connection Manager allows users to maintain multiple logical connections to DB2 for z/OS over a single physical thread. The total number of connections to the database is minimized. Users of the applications may start many logical connections, but these are managed and funneled by Connection Manager into one physical connection per unit of work (UOW).

Process Manager

Process Manager allows multiple users to coexist in a single address space. Each user may have more than one active process. Each user's identity is maintained within the address space. Figure 1, on a high level, the different parts that are cooperating and communicating within Process Manager.

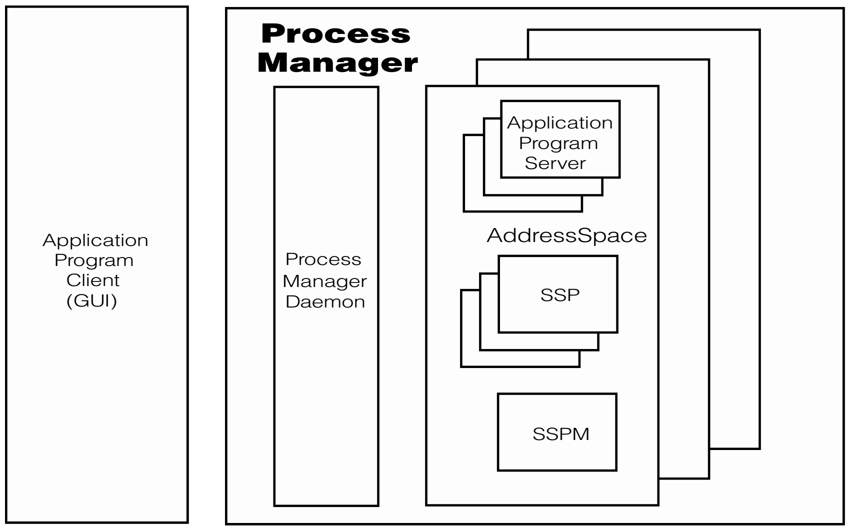

Figure 1. Process Manager overview

Application Program (Client)
: This is a graphical user interface example of an application client, which communicates with the application server through TCP/IP.

Process Manager Daemon
: This is the communication and management focal point within Process Manager.

Server Shell Process Manager (SSPM)
: This is the master process within an address space. SSPM starts and controls the server shell process in its address space.

Server Shell Process (SSP)
These processes are prestarted in several address spaces. Each SSP will execute an application program (server), based on a request from the application program (client).

Application Program (Server)
The application program (server) services requests from the application program (client). There are two types of servers:

Remote Connected Server (RCS)
An application program (server) that is started by a prestarted server shell process.

Local Connected Server (LCS)
An application program (server) that is started by a remote connected server, normally in the same address space.

National language support

z/OS UNIX System Services Connection Scaling supports the following languages:

- English (LANG=C), which is the default
- Japanese (LANG=Ja_JP)

For further information, refer to ″Customizing for Your National Code Page″ in *z/OS UNIX System Services Planning*.

Chapter 2. Post-installation, configuration, and activation

Connection Manager Post-installation, configuration, and activation

NOT Programming Interface information

Note:

This section describes how to post-install, configure, and activate Connection Manager after the SMP/E installation has been completed. Refer to *z/OS Program Directory*, for the SMP/E installation instructions.

Connection Manager Post-installation

The following files and data sets indicate a successful SMP/E installation of Connection Manager:

Hierarchical File System (HFS) Files:
/usr/lpp/cmx/lib/cmxdll
/usr/lpp/cmx/lib/cmxdll.x
/usr/lpp/cmx/include/cmxcli.h
/usr/lpp/cmx/bin/cmxver
/usr/lpp/cmx/samples/cmx.conf
/usr/lpp/cmx/nls/msg/Ja_JP/cmxerror.cat
/usr/lpp/cmx/nls/msg/C/cmxerror.cat

Data sets:
*hlq.*SCMXDBRM
SYS1.SAMPLIB(CMXBIND)

Refer to *z/OS Program Directory* for complete information about how to install Connection Manager.

Rule: If using Process Manager, then the *cmxdll* installed executable file **must** have the program control extended attribute set. Log on as the root user id and issue the following command: **extattr +ps /usr/lpp/cmx/lib/cmxdll**. You must ensure that the ″root″ user has at least read access to the BPX.FILEATTR.PROGCTL resource in the RACF® facility class profile.

The following example shows the RACF commands used to give this permission to the ″root″ user:

```
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(ROOT) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Connection Manager Version verification

The installed Connection Manager version can be queried using the **cmxver** utility located in */usr/lpp/cmx/bin*. Enter the following command to run the **cmxver** executable:

```
/usr/lpp/cmx/bin/cmxver
```

cmxver reads the version information from the binary and writes it to stdout.

Connection Manager Configuration

1. Run CMXBIND, found in SAMPLIB, to bind your chosen DB2 plan name to the Connection Manager DBRMLIB.

2. Edit */etc/profile* and update the following environment variables:
   a. Add the following path to the PATH environment variable:
      ```
      /usr/lpp/cmx/bin
      ```
   b. Add the following path to the LIBPATH environment variable:
      ```
      /usr/lpp/cmx/lib
      ```
   c. Add the following path to the NLSPATH environment variable:
      ```
      /usr/lpp/cmx/nls/msg/%L/%N
      ```

      See “National language support” on page 2 for information regarding %L.
   d. Ensure the following line is **not** commented-out:
      ```
      export_C89_INCDIRS="/usr/include"
      ```
   e. Add */usr/lpp/cmx/include* to this line:
      ```
      export_C89_INCDIRS="/usr/include /usr/lpp/cmx/include"
      ```

      Note: The separator between the INCLUDE directories */usr/include* and */usr/lpp/cmx/include* is a space character.
   f. Ensure the following lines are **not** commented-out:
      ```
      eval "export$(typeset -x | grep "^_C89_" | awk
           '{sub("_C89_","_CC_");printf "%s ",$0}')"

      eval "export$(typeset -x | grep "^_C89_" | awk
           '{sub("_C89_","_CXX_");printf "%s ",$0}')"
      ```
   g. Ensure the following line is **not** commented-out:
      ```
      export_C89_LIBDIRS="/lib /usr/lib"
      ```
   h. Add */usr/lpp/cmx/lib* to this line:
      ```
      export_C89_LIBDIRS="/lib /usr/lib /usr/lpp/cmx/lib"
      ```

      Note: The separators between the LIBRARY directories */lib*, */usr/lib* and */usr/lpp/cmx/lib*) are space characters.
3. Create the Connection Manager configuration file. The */etc/cmx* directory was created when the CMXISMKD job was run during installation. Copy the sample configuration file from */usr/lpp/cmx/samples/cmx.conf* to */etc/cmx/cmx.conf*. You can perform this copy from an rlogin or OMVS shell session, from the ISHELL, or using BPXBATCH. A sample shell command to perform the copy is listed below.
   ```
   cp   /usr/lpp/cmx/samples/cmx.conf   /etc/cmx/cmx.conf
   ```
4. Edit */etc/cmx/cmx.conf* and change the DB2 plan name to match the one created when CMXBIND was run, and the SSID (DB2 subsystem ID) to match your environment. The other values provided in the sample file are adequate for most installations. If you want to modify any of the defaults, refer to “Connection Manager Configuration file”.

Connection Manager Configuration file

Connection Manager includes one configuration file that resides, by default, in */etc/cmx/cmx.conf*. The location and name of this file can be set with the *$CMXCONF* environment variable by specifying:

```
export CMXCONF=/etc/cmx/mystuff/mycmx.conf2
```

The following are parameters configured in *cmx.conf*:

DB2SubSystem=yyyy
    The name of the DB2 subsystem.

DB2PlanName=xxxxxxxx
: The DB2 plan name. The DB2 BIND job, **(CMXBIND)**, runs after Connection Manager installation and creates this plan from the shipped DBRMLIB.

DB2Attach=CAF | RRS
: The attachment method used by Connection Manager to connect to DB2 for z/OS. The choices are either CAF or RRS. The default is RRS.

DB2ccsid=nnn
: The DB2ccsid parameter allows you to build an ASCII database (specified in the **ENSCHEME DSNDECP** parameter during DB2 subsystem build). The benefit is that an ASCII application's data can be stored in ASCII format, accessed from the application in readable format, and accessed in readable format with other DB2 for z/OS utilities, such as SPUFI. This is the recommended method of operation for ASCII-based applications.

It is very important that the **DB2ccsid** parameter be set to the same type of code page that was used when creating and loading the database. If the ccsids are different, data translation may occur, and this can degrade performance.

For ASCII databases, specify ASCII in the **ENSCHEME** parameter of DB2 DECP setup, and specify the ASCII ccsid of the codepage you want in the *asccsid* parameter of DECP. Then specify the same ccsid in the **DB2ccsid** parameter of the *cmx.conf* file. (The default ASCII code page is 819.)

For EBCDIC databases, specify EBCDIC in the **ENSCHEME** parameter of DB2 DSNDECP setup, and specify the ccsid of the EBCDIC codepage you want in the **sccsid** parameter of DSNDECP. Then specify the same ccsid in the DB2ccsid parameter of the */etc/cmx/cmx.conf*. (The default EBCDIC code page is 37.)

As a performance option for EBCDIC databases only, you can specify a DB2ccsid of 0. This will disable any ccsid support. Note that this is valid only if you have an EBCDIC database. In this case only, character data is still stored in tables as ASCII, but no checking for translation will occur on fetch, update, or insert. As a result, DB2 utilities will not see returned character data translated to EBCDIC.

DB2datefmt=ISO| USA | EUR | JIS
: Specifies the DB2 on z/OS ZPARM for DATE datatype format. The default, when not specified is ISO.

Trace=0 | 1
: Turns Connection Manager trace on (1) and off (0). The default is 0. See "User tracing" on page 37.

TraceWrap=nnnnn
: The maximum number of trace lines written to each user's trace file before it wraps. TraceWrap=0 implies no wrapping of trace. The default is 1000 lines.

TraceShort=1 | 0
: The SQL statement traced on a SQL prepare will be truncated to 136 characters if TraceShort = 1. The default is 0, which causes the entire SQL statement to be output on a SQL prepare.

TraceTables=zzzzzzzz
: Specifies a list of tables that should be traced in detail. The list is a set of one or more table names separated by commas. The detail trace will format

all input and output SQLDA variables for the specified list of tables. It is imbedded in the normal summary trace file, *cmxtrace.'userid'*, and is only enabled when Trace=1. The default is no detailed table tracing.

TraceUser=ALL | username | path/file
Specifies tracing for all users, 1 user, or a list of users in the file specified. This parameter is mandatory if you specify Trace=1. No default is provided.

DB2ErrorLog=1 | 0
Turns Connection Manager error logging on (1) and off (0) for all negative SQL codes. The default is 0.

DB2IgnoreErr=nnnn
When DB2ErrorLog has been set to 1(on), this parameter specifies a list of one or more negative SQL codes that will cause these particular errors to not be logged by Connection Manager. For example, a duplicate row error, -803, can be put in this list so that application handling of a duplicate row will be transparent in the Connection Manager error log.

The information below is an example of a Connection Manager cmx.conf file. The following values are used in the example:

PDB1	DB2 for z/OS subsystem name
CMXPLN	Plan name specified at BIND time
819	ASCII code page used to load data

```
//--------------------------------------------------------
// Connection Manager Configuration File
//--------------------------------------------------------

[NAMES]
DB2SubSystem=PDB1
DB2PlanName=CMXPLN
DB2Attach=CAF

[CCSID]
DB2ccsid=819
DB2datefmt=ISO

[TRACE]
Trace=0
TraceWrap=0
TraceTables=Table1,Table2, Table3
TraceUser=ALL
TraceShort=0

[ERROR]
DB2ErrorLog=1
DB2IgnoreErr=-803
```

Connection Manager activation

Connection Manager consists of a set of interfaces and is not activated until an application is linked with the provided interfaces. More information about required application changes and the Connection Manager interface can be found in "Connection Manager Application program interface" on page 25.

The following are 2 environment variables that may affect Connection Manager performance. These environment variables should be set in the .profile of each user using these features. The description and syntax are as follows:

CMX_THREAD_WAIT
Whenever an adjunct thread commits or rolls back a transaction, it will wait for CMX_THREAD_WAIT seconds before being removed and detached. If

more work occurs for the adjunct thread prior to this expiration time, the thread remains active and the timer is reset until the next COMMIT/ROLLBACK.

If CMX_THREAD_WAIT is set to 0, the adjunct threads will be removed upon COMMIT/ROLLBACK. If CMX_THREAD_WAIT is unset, the default of 10 seconds will be used. For applications where there are frequent multiple COMMIT scopes outstanding, CMX_THREAD_WAIT will reduce thread create/destroy overhead. For example:

```
export CMX_THREAD_WAIT=20
```

sets the thread wait time to 20 seconds.

CMX_CURSOR_SHARE

This allows identically prepared statements across different logical user sessions to be shared. The cursor position is not maintained for all sessions simultaneously, so extra OPENS may be required of the application.

However, it does minimize the total open cursors in DB2 on z/OS, and the attendant CPU and storage overhead. The default is 0 (OFF). For example:

```
export CMX_CURSOR_SHARE=1
```

sets on cursor sharing mode.

For applications which use common SQL across sessions, CMX_CURSOR_SHARE will reduce the total number of DB2 on z/OS cursors.

End of NOT Programming Interface information

Process Manager Post-installation, configuration, and activation

NOT Programming Interface information

Note:

This section describes how to post-install, configure, and activate Process Manager after the SMP/E installation has been completed. Refer to *z/OS Program Directory*, for the SMP/E installation instructions. This section assumes that Process Manager SMP/E installation has been performed successfully.

Process Manager Post-installation

The Process Manager files reside in the z/OS hierarchical file system (HFS). The following files and directories, created in an HFS, indicate a successful SMP/E installation:

/usr/lpp/bpa/bin/bpadaemn
/usr/lpp/bpa/bin/bpalogd
/usr/lpp/bpa/bin/bpassp
/usr/lpp/bpa/bin/bpasspm
/usr/lpp/bpa/bin/bpastartgc
/usr/lpp/bpa/bin/bpaver
/usr/lpp/bpa/samples/post_inst_bpa
/usr/lpp/bpa/bin/p_dae
/usr/lpp/bpa/bin/s_dae
/usr/lpp/bpa/bin/s_gc

/usr/lpp/bpa/include/bpaunix.h
/usr/lpp/bpa/lib/bpadll
/usr/lpp/bpa/lib/bpadll.x
/usr/lpp/bpa/samples/bpares.cfg
/usr/lpp/bpa/nls/msg/Ja_JP/bpaerror.cat
/usr/lpp/bpa/nls/msg/C/bpaerror.cat

Post-installation procedure

1. Log in as a root user.

   Rule: The installed executable files ″bpassp″, ″bpasspm″, and ″bpadll″ **must** have extended attribute ″program-controlled″ set by the command **extattr +ps**, which requires that the user has permission to do so. You must ensure that the root user has at least read access to the BPX.FILEATTR.PROGCTL resource in the RACF facility class profile.

   The following example shows the RACF commands used to give this permission to the root user:

   ```
   RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
   PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(ROOT) ACCESS(READ)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

   Programs retrieved from libraries that are not designated as program controlled, mark the address space in which they execute as dirty. To ensure multiple processes can be started within a single address space, you must issue the RALTER RACF command with option NOPADCHK, to mark selected libraries as program controlled. Include LINKLIB (SYS1.LINKLIB), C runtime library (SYS1.SCEERUN), TCP/IP libraries (SYS1.SEZATCP and SYS1.SEZALOAD), and all DB2 SDSNLOAD, SDSNEXIT, and SDSNLINK libraries. For example:

   ```
   RALTER PROGRAM ** ADDMEM('SYS1.LINKLIB'/'******'/NOPADCHK  +
          'SYS1.SCEERUN'/'******'/NOPADCHK  +
          'SYS1.SEZATCP'/'******'/NOPADCHK  +
          'SYS1.SEZALOAD'/'******'/NOPADCHK  +
          'hlq.SDSNLINK'/'volser'/NOPADCHK  +
          'hlq.SDSNLOAD'/'volser'/NOPADCHK  +
          'hlq.SDSNEXIT'/'volser'/NOPADCHK) UACC(READ)
   ```

2. Complete the Process Manager post-installation procedure by entering:

   ```
   /usr/lpp/bpa/samples/post_inst_bpa
   ```

   A step within the shell script *post_inst_bpa* copies the file *bpares.cfg* from */usr/lpp/bpa/samples* to the */etc/bpa* root directory. The */etc/bpa* directory was created when the CMXISMKD job was run during the install procedure. You must change the IP address and port number in the */etc/bpa/bpares.cfg* file to match your environment. The other values provided in the sample */etc/bpa/bpares.cfg* file are adequate for most installations. If you want to modify any of the defaults, refer to “Process Manager configuration file” on page 9.

Process Manager Version verification

The installed Process Manager version can be queried using the utility located in */usr/lpp/bpa/bin*. Enter the following command to run the **bpaver** utility:

```
/usr/lpp/bpa/bin/bpaver
```

The **bpaver** utility reads the version information from all binaries and writes it to stdout.

Process Manager configuration

This section describes the Process Manager configuration parameters in detail. You should have a working knowledge of z/OS, because Process Manager manages processes in z/OS address spaces.

1. Edit */etc/profile* and update the following environment variables:
   a. Add the following path to the PATH environment variable:
      ```
      /usr/lpp/bpa/bin
      ```
   b. Add the following path to the LIBPATH environment variable:
      ```
      /usr/lpp/bpa/lib
      ```
   c. Add the following path to the NLSPATH environment variable:
      ```
      /usr/lpp/bpa/nls/msg/%L/%N
      ```

      See “National language support” on page 2 for information regarding %L.
2. As a result of the “Post-installation procedure” on page 8, */usr/lpp/bpa/samples/bpares.cfg* is copied to */etc/bpa/bpares.cfg*. Process Manager reads the configuration data from */etc/bpa/bpares.cfg*.
3. If you use the supplied sample configuration file, you must at least change the IP-address and the port number in the configuration file provided. The other values provided in the sample */etc/bpa/bpares.cfg* file are adequate for most installations. See “Process Manager configuration file”.
4. If a system internal process will use the Process Manager to execute programs under its control, the capability to listen on those system internal requests may be enabled by exporting a environment variable:
   ```
   export _BPA_IPC_BASE_PATH=<existing valid directory>
   ```

   prior to startup of Process Manager as well the startup of the requestor process. As example use the same as the BasePath variable */var/bpa*. This environment variable is used as communication point for both processes. The sample script *s_dae*, which may be used to chain the start-up of the requester process with Process Manager processes startup, is prepared with comments to export the environment as well as other processes which are communicating.

Process Manager configuration file

Process Manager uses configuration file */etc/bpa/bpares.cfg* to define all parameters needed to run Process Manager.

Rule: The following parameters **must** be set before start-up:

```
[DAEMON] IP-Address=<the ip-address of your system>
[DAEMON] Port=<any valid TCP/IP port number>
```

Rule: The following parameters **must** be changed and point to existing valid directories before start-up:

```
[BASE] BasePath=/var/bpa
[BASE] ErrorLog=/var/bpa/bpaerr.log
[TRACE] KeyFileName=/var/bpa/bpalogd.key
[TRACE] LogFileName=/var/bpa/bpatrace.log
```

Directory */var/bpa* is only necessary if you follow the suggestions made in the default Process Manager configuration file. This directory will be used to store ″keyfiles″, z/OS UNIX domain sockets, trace logs, and error logs. It can be removed if you make appropriate changes to the default configuration.

The configuration file is divided into ***Groups***, which are enclosed in brackets. A group combines a set of parameters. C++-like ″//...″ comments are allowed within the configuration file. The following information is an example of a Process Manager configuration file:

```
//-----------------------------------------------------------------------------
// Process Manager Configuration File (bpares.cfg)
//-----------------------------------------------------------------------------

[BASE]
BasePath=/var/bpa
ErrorLog=/var/bpa/bpaerr.log

[SYSTEM]
MaxAddressSpaceSize=1000000
MaxTCBPerAS=50

[DAEMON]
IP-Address=9.164.156.162
Port=3888
RestartToShutdownThreshold=10
RestartDelayTime=10
LITQueueLength=500
MaxThreadTasks=100
MaxThreads=100
ThreadStackSize=32

[SSPT]
SSPWaitTime=600
TrimmTime=300
MaxRestartSSPM=10
PrestartedAS=1
PrestartedSSPPerAS=2
MaxSSPPerAS=4
MinFreeSSP=1
MaxFreeSSP=4
StartAddSSP=2
TermAddSSP=2
MaxMemBelowPerUser=350
MinFreeMemBelowPerAS=40

[TRACE]
Level=1
FileRecords=500
BufRecords=500
KeyFileName=/var/bpa/bpalogd.key
LogFileName=/var/bpa/bpatrace.log
ToStdout=0                         // No output to stdout

[USER]
OLGA=1   // Optional - Userid OLGA, long-running jobs with high memory consumption.
MARK=1  // Optional - Userid MARK, long-running jobs with high memory consumption.
JOE=0     // Optional - Userid JOE, no longer has long-running jobs.
[SHARED_MEMORY]
SharedMemSupport=0
```

Process Manager scenarios

The following sections describe some of the most common scenarios. An understanding of these scenarios is helpful to the system operator who has to modify default values in the Process Manager configuration file.

Starting the Process Manager daemon: When starting the Process Manager daemon, a specific number of server shell processes are started. This number has to be specified in the configuration file by the following parameters:

```
[SSPT]
PrestartedAS=2
PrestartedSSPPerAS=2
```

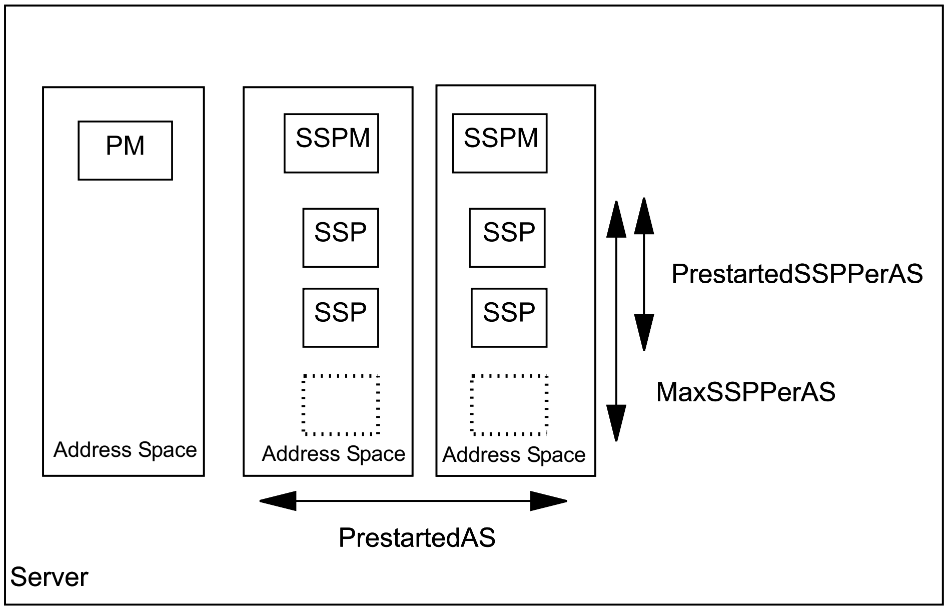

Figure 2. Starting the Process Manager daemon

With the parameter *MaxSSPPerAS* the system operator specifies the maximum number of server shell processes which should be started within an address space. Each process executes a remote connected server program when a client application sends a request. The number of *MaxSSPPerAS* equals the number of users, executing within one address space.

Trimming: Process Manager controls the free server shell processes. Periodically Process Manager checks the status of the server shell processes to determine if there are a sufficient number of processes free to enable assigning of new work or if idle processes need to be terminated. This balancing is referred to as positive or negative trimming. Those limits, when new server shell processes have to be started or terminated is configurable by parameters within the configuration file.

Positive trimming: Process Manager performs positive trimming whenever there are not enough free server shells running using the following parameters in the configuration file:

```
[SSPT]
MinFreeSSP=2
StartAddSSP=4
```

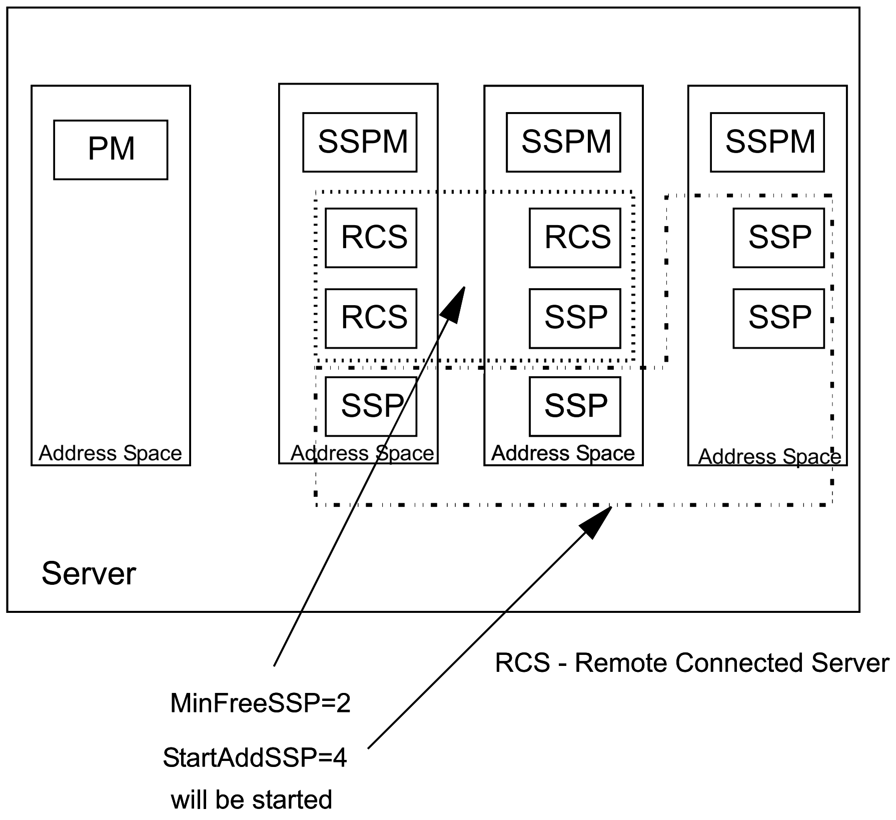

Figure 3. Positive trimming

Whenever the number of currently free server shell processes is less than specified in *MinFreeSSP*, an additional number (StartAddSSP) of Server Shell Processes is started. If all started address spaces are running with their specified number of MaxSSPPerAS, positive trimming will start new address spaces.

Negative trimming: Process Manager performs negative trimming whenever too many free server shell processes are running using the following parameters in the configuration file:

```
[SSPT]
MaxFreeSSP=2
TermAddSSP=4
```

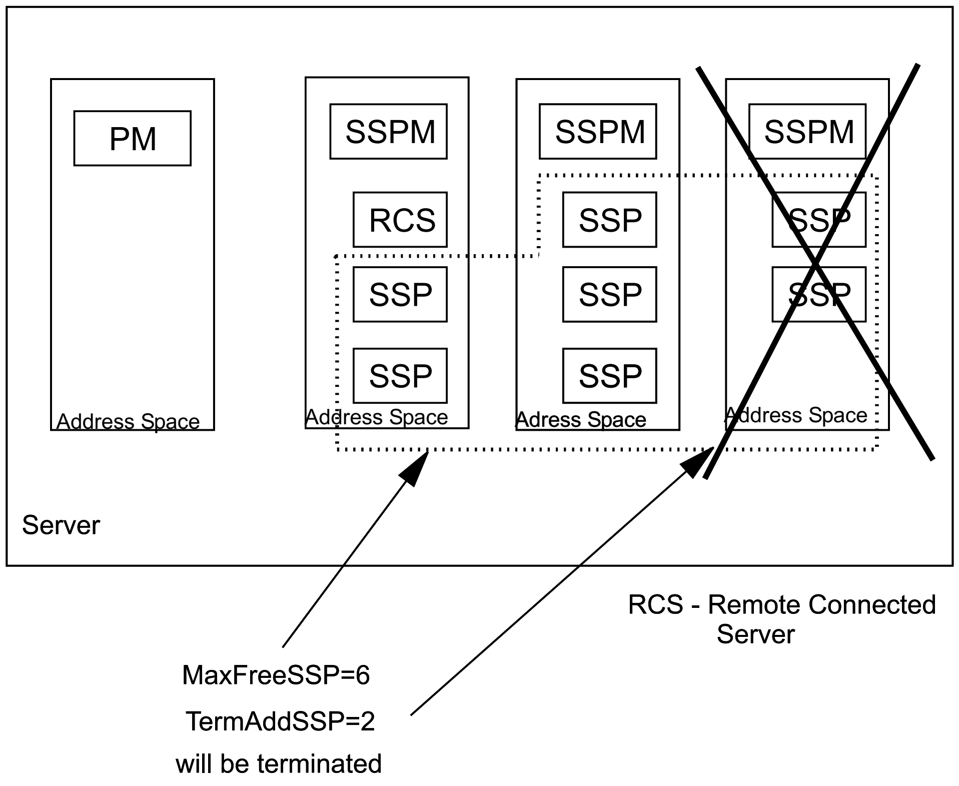

Figure 4. Negative trimming

Process Manager terminates address spaces which do not contain any server shell processes.

Trimming on demand: When many users connect at the same time (high system load), Process Manager may not be able to reserve or prestart enough server shell processes to handle all those requests concurrently. This situation, referred to as (positive) trimming on demand, forces Process Manager to start additional server shell processes. The client which tries to connect to the server will have to wait until the new started server shell process becomes available. The resource parameter SSPWaitTime specifies the maximum wait time. If the specified time expires, the client's request is rejected.

MaxTCBPerAS: Whenever a remote connected server process wants to start a new process it uses the function bpa_spawn(), which is an adaptation (fake) of the spawn() function provided by the operating system. This function starts new processes, either into the local address space or as an exception in a new address space.

The parameter MaxTCBPerAS defines the trigger for this decision. In this context, the number of TCB's is equal to the number of processes (Number_of_TCB = OFFSET + Number_of_processes), where OFFSET is a small number of TCB's (about 5) which are always started.

If the actual number of TCB's is less than this defined threshold the Process Manager_spawn() function will try to start the new process in the same address space otherwise it will start it into a new address space.

```
 bpa_spawn()
{
    ...
    if (actual number of running TCBs < MaxTCBPerAS)
    {
        rc = spawn(process, local)
        if (rc = bad return code because of missing resources)
        {
            rc = spawn(process, foreign)
        }
    } else
    }
        rc = spawn(process, foreign)
    }
    ...
}
```

Note: Lower the value for this parameter only if you observe problems with the spawn() function.

SSPWaitTime: This parameter specifies, how long a client will wait to get a SSP assigned, which will execute the specified process (remote connected server). If the SSPWaitTime ends, the rexec request will time out (login fails).

The measurement of SSPWaitTime starts at the moment, where the client application rexec request is received by Process Manager. It ends at the moment, Process Manager has successfully assigned a server shell process to this request, or when the value specified within the configuration file is reached.

A timeout may occur if the SSPT has to start additional SSPs for a new client request (positive trimming on demand). The client process has to wait for this new Server Shell Process to become available.

Note: If small numbers are specified for *MinFreeSSP* and *StartAddSSP* this situation might appear more frequently.

TrimmTime: This parameter specifies, how long the SSPT is waiting for a response message from a new started server shell process. The counting of the actual TrimmTime starts, whenever new server shell processes have to be started (either during normal positive trimming or because of a need for trimming on demand).

If the value specified in the configuration file for this parameter is reached before the new started Server Shell Process becomes available, Process Manager assumes a lost message and tries to start once more new Server Shell Processes.

If the value for this parameter is too low, and we have a high system load on the server machine, the message from a successfully started new server shell process may not be received within time. This leads to the situation, that Process Manager tries to start even more server shell processes. When all response messages for all the new server shell processes arrive, Process Manager can be forced to decide to terminate now some of them (negative trimming). In other words, the whole system becomes relatively ″nervous″. It starts and terminates SSPs often, which leads to a higher load and poor response time.

MaxSSPPerAS: This number should be as high as possible, to archive the goal of minimizing the total number of address spaces consumed by server processes executed for the clients applications. However, too many processes within one address space could cause each new local connected server to be spawned into a new address space. The SSPT does not control these new address spaces and these address spaces are used exclusively by the spawned process.

The value specified here will be checked during daemon start-up. The daemon checks the maximum available memory below the line and compares it with the memory consumption below the line of server applications started for one client application.

MaxMemBelowPerUser / MinFreeMemBelowPerAS:

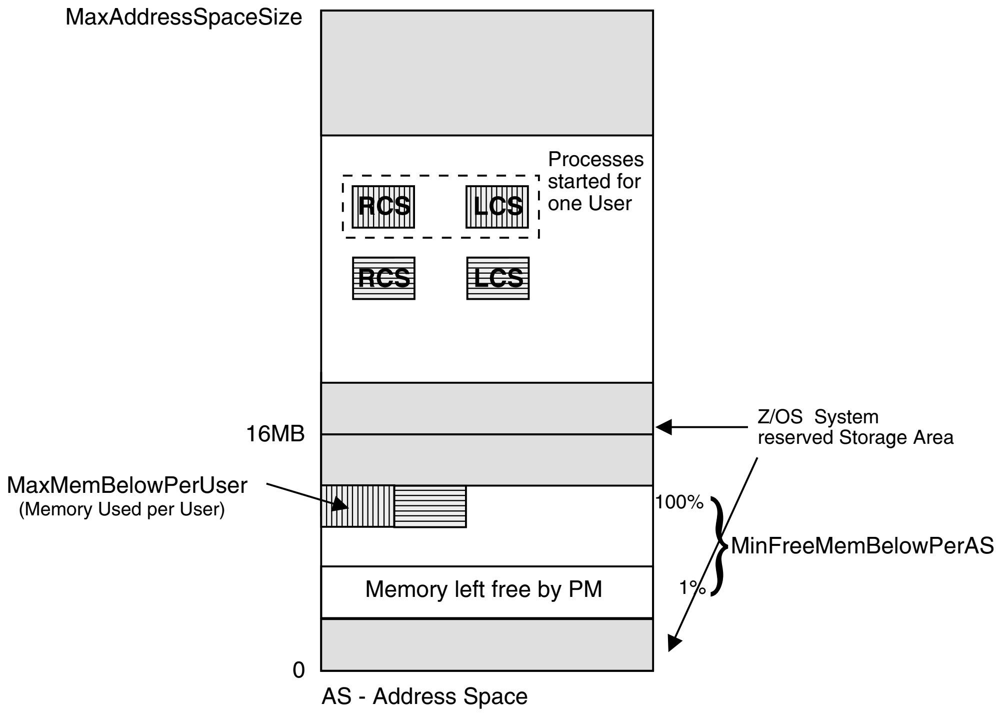

Figure 5. Memory usage per user within one address Space

MaxMemBelowPerUser and *MinFreeMemBelowPerAS* are used to calculate how many SSPs can be started within one address space. *MaxMemBelowPerUser* is the amount of storage, typically consumed by all server processes, started for a normal, expected client application (in other words, for all processes started on the server side for one user). This value is difficult to specify. Please ask your IBM Service partner to assist you in determining an optimum value (we suggest an initial value of *MaxMemBelowPerUser*=350 and *MinFreeMemBelowPerAS*=40).

The *MinFreeMemBelowPerAS* is a reserve. If your client processes will never consume more memory below the 16MB boundary, you can lower the value specified here.

Optional group USER: Some server applications started for one client application might have more resource requirements (mainly memory) than the average required for other processes. To avoid termination of those processes because of resource shortage when running concurrently, it is possible to start these processes in an exclusive owned address space controlled by Process Manager.

This behavior can be forced by adding the userid of the person starting such processes to the (optional) group USER.

Note: Consider creating a dedicated userid for this purpose.

Process Manager Configuration file parameters: If you have special requirements, and cannot use the sample values provided, this section supplies detail on the meaning and value ranges for all parameters within the configuration file. Parameters are arranged in the following tables by group.

Notes:

1. All parameters indicated below as required must be specified; no default settings are provided.
2. If a required parameter (see column “Req”) is missing or assigned an invalid value, Process Manager will not start.
3. If you do not provide a value for an optional parameter, either the system settings specified in the BPXPRMxx member are implicitly used or the default setting shown below is used.

Table 1. BASE parameters. These entries are used to configure basic values.

Req	Entry	Range	Default	Description
Yes	BasePath	Valid absolute pathname	None	BasePath determines the base path prefixed to socket paths used in context of Process Manager.
Yes	Errorlog	Valid absolute pathname	None	Errorlog determines the pathname of the error log file used by Process Manager.

Table 2. SYSTEM parameters. These entries are used to configure system parameters.

Req	Entry	Range	Default	Description
No	MaxAddressSpaceSize	1,000,000 to 2,000,000	MAXASSIZE in BPXPRMxx	This entry determines the maximum Address Space size for the process started by the daemon (SSPM) as a multiple of 1K (1024) bytes (″softlimit″). Refer to *z/OS MVS Initialization and Tuning Reference*.
No	MaxTCBPerAS	10 to 1000	50	This entry determines the maximum number of TCBs running in one address space. If this number is exceeded, additional processes are started in a new address space.

Table 3. DAEMON parameters. These entries are used to configure the Process Manager daemon.

Req	Entry	Range	Default	Description
Yes	IP-Address	a.b.c.d a.b.c a.b a	None	This entry determines the host address used by the listener thread of the Process Manager daemon. It must be specified in standard dotted-decimal notation. If ″IP-Address=0″ is specified, the Process Manager daemon is bound on all network interfaces on the host and listens on the same port.
Yes	Port	1 to 99999	None	This entry determines the port to which the Process Manager daemon listener thread must bind.
No	RestartToShutdownThreshold	0 to 1000	10	This entry determines the number of restarts tried for the TCP/IP communication threads of the Process Manager daemon.
No	RestartDelayTime	0 to 3600	60	This entry determines the delay time, in seconds, before restarting a thread of the Process Manager daemon. See parameter RestartToShutdown Threshold for more information.
No	LITQueueLength	10 to 9999	1000	This entry determines the maximum length of the queue (or the number of pending connections) for the Process Manager daemon listener thread.
No	MaxThreadTasks	50 to 32768	1000	This entry determines the softlimit of the maximum TCBs available to the process for concurrently running threads. The Process Manager daemon runs one listener thread, dispatching each incoming login request exclusively to one login handler thread. MaxThreadTasks can be used to specify how many login handlers can operate in parallel.

Table 3. DAEMON parameters (continued). These entries are used to configure the Process Manager daemon.

Req	Entry	Range	Default	Description
No	MaxThreads	200 to 100,000	1000	This entry is the softlimit of the maximum number of threads a process can run concurrently. The Process Manager daemon runs one listener thread, dispatching each incoming request exclusively to one login handler thread. MaxThreads can be used to specify how many login handler threads can be used in parallel. However, the maximum number of threads share the maximum number of threadtasks, MaxThreadTasks.
No	ThreadStackSize	1 to 100,000	10	This entry determines the initial thread stack size, as a multiple of 1K (1024) bytes. Other stack characteristics, like stack increment size, are inherited from the STACK _CEE_RUNOPTS run-time option.

Table 4. SSPT parameters. These entries are used to configure the behavior of the server shell process table.

Req	Entry	Range	Default	Description
No	MaxRestartSSPM	0 to 1000	10	Maximum number of terminating SSPMs with bad return codes. If more SSPMs failed in execution, this case will be logged and no attempt will be made to restart the SSPM.
No	SSPWaitTime	1 to 3600	60	This parameter specifies how long a client will wait to get a SSP assigned, which will execute the specified process (remote connected server). This is actually the login timeout and should be greater than TrimmTime. This value is specified in seconds.
No	TrimmTime	1 to 3600	30	This parameter specifies how long the process manager daemon waits for a response from an SSP. This value is specified in seconds.
Yes	PrestartedAS	1 to 1000	None	The number of prestarted address spaces (each controlled by an SSPM) after Process Manager daemon start-up.

Table 4. SSPT parameters (continued). These entries are used to configure the behavior of the server shell process table.

Req	Entry	Range	Default	Description
Yes	PrestartedSSPPerAS	1 to 100	None	The number of SSPs which are prestarted by any new SSPM in the same address space.
Yes	MaxSSPPerAS	1 to 100	None	The maximum number of SSPs running in one address space.
Yes	MinFreeSSP	1 to 1000	None	The minimum number of all server shell processes controlled by the Process Manager daemon, which must be available to assign new work. When this threshold is reached, a number of `StartAddSSP` new server shell processes will be started.
Yes	MaxFreeSSP	1 to 1000	None	The maximum number of free server shell processes controlled by the Process Manager daemon. When this threshold is reached, `TermAddSSP` free SSPs will be stopped.
Yes	StartAddSSP	1 to 1000	None	The number of SSPs to be started when `MinFreeSSP` threshold is reached.
Yes	TermAddSSP	1 to 1000	None	The number of SSPs to be stopped when `MaxFreeSSP` threshold is reached.
No	MaxMemBelowPerUser	1 to 10,000	500	Describes the maximum memory consumed below 16 MB by an average user. This value is used to determine at start up, how many SSP's (users) will fit into an address space. See Table 8 on page 21 for a description of how this value results in an adjustment of the `MaxSSPPerAS` value. You can obtain this value by determining, during run-time tests, how much memory below the 16 MB boundary is consumed by a typical client user executing remote connected server processes under the control of the Process Manager daemon. These two parameters are used to calculate how many SSPs can be started within one address space. The `MinFreeMemBelowPerAS` value is like a reserve. If you are confident that your client processes will never consume more memory below the 16 MB boundary, you can lower the value specified here.

Table 4. SSPT parameters (continued). These entries are used to configure the behavior of the server shell process table.

Req	**Entry**	**Range**	**Default**	**Description**
No	MinFreeMemBelowPerAS	1 to 100	50	This entry describes the percentage of total available storage below the 16 MB boundary that must be kept free (for unexpected usage exceeding the value specified in `MaxMemBelowPerUser`).

Table 5. TRACE parameters. These entries are used to configure tracing.

Req	**Entry**	**Range**	**Default**	**Description**
Yes	Level	0 to 3	None	Specifies the trace level: 0 - trace is switched off 1 - module activation and deactivation will be traced 2 - additional information trace is active 3 - trace level 1 and 2 will be activated together
Yes	FileRecords	1 to 2,000,000	None	The maximum number of trace records saved to file.
Yes	BufRecords	0 to 10,000	None	The number of trace records the buffer should hold before data will be appended to the configured trace log file. **Note:** Specifying a value of 0 will result in trace data being written to file immediately; but this is HFS I/O intensive. Use **kill -USR1 <bpalogd_pid>** to write this buffer to disk whenever possible.
Yes	KeyFileName	Valid absolute path name	None	Key file name used to get the key to create the message queue. **Note:** All clients must configure this in their configuration file to enable the attachment to the message queue.
Yes	LogFileName	Valid absolute path name	None	Trace log path and file name. **Note:** The client processes read this from the environment.
Yes	ToStdout	1 or 0	None	If set to 1, the trace strings will be dumped to stdout **Note:** This will enable direct trace feedback without using **tail -f tracelog** and is independent from parameter *BufRecords*

Table 6. USER parameters (optional). You can optionally add users (userids) you wish to assign to an exclusive address space (equivalent to an SSPM). A value of "1" enables the user behavior described in this table.

Req	Entry	Range	Default	Description
No	OLGA	0 or 1	0	Userid OLGA executes long-running jobs with high memory consumption if the parameter value equals 1.
No	MARK	0 or 1	0	Userid MARK executes long-running jobs with high memory consumption if the parameter value equals 1.
No	JOE	0 or 1	0	Userid JOE executes long-running jobs with high memory consumption if the parameter value equals 1.

Table 7. SHARED_MEMORY parameters (optional)

Req	Entry	Range	Default	Description
No	SharedMemSupport	0 or 1	1	0 = No shared memory support. The SSPM does not attach shared memory to the address space. 1 = The SSPM attaches the shared memory to a specific address of the address space during startup.

Plausibility checks of configuration parameters: All optional configuration parameters have default values. During start-up of the Process Manager daemon, a semantic check is performed on all parameters for which you have provided values. Some values may be modified if they are not consistent with other parameters or if current system resource parameters are exceeded.

The following checks are performed and the configuration parameters are adjusted as described in Table 8. Checks are performed in the sequence shown.

Table 8. SSPT parameter checking. This table describes the checks performed on the values specified for the parameters in the group SSPT.

No	Check	Reaction
1	MaxSSPPerAS > (StorageBelow16M - MinFreeMemPerAS) / MaxMemBelowPerUser	MaxSSPPerAS = (StorageBelow16M - MinFreeMemPerAS) / MaxMemBelowPerUser
2	PrestartedSSPPerAS > MaxSSPPerAS	PrestartedSSPPerAS = MaxSSPPerAS
3	MaxFreeSSP < StartAddSSP	StartAddSSP = PrestartedSSPPerAS
4	MaxFreeSSP < TermAddSSP	TermAddSSP = StartAddSSP
5	MaxFreeSSP ≤ MinFreeSSP	MaxFreeSSP = MinFreeSSP + StartAddSSP
6	MaxFreeSSP < (PrestartedAS * PrestartedSSPPerAS)	MaxFreeSSP = (PrestartedAS * PrestartedSSPPerAS)

Process Manager activation

This section covers the following topics:

- Preparing to start Process Manager
- Starting Process Manager without using JCL

- Starting Process Manager using JCL
- Shutting down Process Manager
- Starting the Garbage Collector

Preparing to start Process Manager

Before you begin: You must have customized the configuration file before start-up.

1. Log in as a superuser (root), with a UID of 0.

   Rule: This user must have read access to the BPX.DAEMON RACF profile in FACILITY class.

   The following example shows the RACF commands used to give this permission to the root user:

   ```
   RDEFINE FACILITY BPX.DAEMON UACC(NONE)
   PERMIT BPX.DAEMON CLASS(FACILITY) ID(ROOT) ACCESS(READ)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```
2. Check that the file system which contains the directory ″/usr/lpp/bpa″ is mounted without NOSETUID in effect. Also check this on file systems containing other executables and other DLLs that run under the control of Process Manager. See the following **mount** command as example:

   ```
   MOUNT FILESYSTEM('HFS.BPADS') MOUNTPOINT('/usr/lpp/bpa') TYPE(HFS)
   ```

   Note: You must not use NOSETUID, which prevents the use of setuid, setgid, APF and program control attributes. Process Manager relies on having the attribute program control in effect. If NOSETUID is used, the address space controlled by Process Manager is flagged ″dirty″ when an image of an executable is loaded into this address space after successful log in. Subsequent login requests are not permitted to processes in dirty address spaces.
3. Prepare the Process Manager environment. Export the following variables using shell commands:

   ```
   export PATH="/usr/lpp/bpa/bin:/usr/lpp/bpa/samples:${PATH}"
   export LIBPATH="/usr/lpp/bpa/lib:${LIBPATH}"
   export NLSPATH="/usr/lpp/bpa/nls/msg/%L/%N:${NLSPATH}"
   ```
4. Prepare the environment as needed to run the full set of applications. For example, export the application program variables, start the application program shared memory manager, and so on.
5. Check that all application program executables and DLLs that might be loaded into the storage of address spaces controlled by Process Manager have the extended attributes set by the **extattr +ps** command. Identify the application program executables to be run by Process Manager.
6. All users will inherit the environment from Process Manager. If you have specific user requirements, define the appropriate environment in the *bpaprofile* located in the user’s home directory. In a *bpaprofile* you can specify an environment variable by using *<variable-name>*=*<value-string>* in one line. This variable is active in the environment of each process owned by the user and controlled by Process Manager.

   Note: You must not use quotes with <value-string> if it is not part of the value.

   If the home directory is shared among more than one user, the *bpaprofile* is shared also. For example:

```
_CEE_RUNOPTS=HEAP(17M,1M,ANYWHERE,KEEP),
             STACK(131072, 131072,ANYWHERE,KEEP),
             ALL31(ON),
             LIBSTACK(8),
             TERMTHDACT(UADUMP)
```

Starting Process Managerwithout using JCL

After preparing to start Process Manager, enter **s_dae** to execute the start-up script.

Starting Process Manager using JCL

After preparing to start Process Manager, you can start Process Manager using JCL that runs BPXBATCH, which in turn runs the */usr/lpp/bpa/bin/s_dae* shell script.

Processes started by BPXBATCH do not gather code page configuration data from a controlling terminal; this is done by processes started as a background process in a login shell (see “Preparing to start Process Manager” on page 22). This code page information is required to translate input/output data between an ASCII client that is connected through the Process Manager to the EBCDIC environment.

To enable ASCII-to-EBCDIC code page translation, you must export two additional environment variables while submitting BPXBATCH:

- _BPA_CP_ASCII=ascii_codepage
- _BPA_CP_EBCDIC=ebcdic_codepage

Note: Using the correct code page configuration in BPXBATCH for Process Manager start-up is required for proper code page translation function. You can add other environment variables as well.

To find out the active code page configuration, you can do one of the following:

- Check the UNIX System Services shell environment configuration
- Use the Process Manager trace log which is written by Process Manager in [TRACE] Level=3, when Process Manager is started using the shell script as a login shell. For more information, see “Process Manager Configuration file parameters” on page 16.

For example, to start Process Manager using JCL, perform the following steps:

1. Change the */etc/bpa/bpares.cfg* file to show [TRACE] Level=3. The default is Level=0. This change enables tracing and will print the active code pages.
2. Use the */usr/lpp/bpa/bin/s_dae* shell script to start Process Manager.
3. Stop Process Manager using the *p_dae* shell script
4. Run this command:
   ```
   grep ASCII /var/bpa/bpatrace.log
   ```
   This should indicate the active code pages.
5. Create the BPXBATCH JCL, including the code page environment variables. See the sample JCL below.
6. To enable the *s_dae* shell script to be executed by BPXBATCH, modify the last line of the script */usr/lpp/bpa/bin/s_dae* by removing the &. Change:
   ```
   bpadaemn /etc/bpa/bpares.cfg &
   ```
   to
   ```
   bpadaemn /etc/bpa/bpares.cfg
   ```

Here is the sample JCL:

```
//RWBRUPM  JOB MSGCLASS=X,NOTIFY=RWBRU,REGION=4096K,MSGLEVEL=(1,1),
//         CLASS=A,USER=XXXXX,PASSWORD=zzzzzz
//*-----------------------------------------------------------------*//
//* FUNCTION: JCL Job to start Process manager via BPXBATCH          *//
//* CHANGE ACTIVITY:                                                 *//
//*   20-Jun-01 WB create                                            *//
//*-----------------------------------------------------------------*//
//*-----------------------------------------------------------------*//
//*
//STARTPM  EXEC PGM=BPXBATCH,
//         PARM='SH /usr/lpp/bpa/bin/s_dae',
//         TIME=NOLIMIT,REGION=4096K
//*
//STDOUT   DD PATH='/tmp/pmstart.out',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//STDERR   DD PATH='/tmp/pmstart.err',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//STDENV   DD *
_BPX_BATCH_SPAWN=YES
_BPA_CP_ASCII=ISO8859-1
_BPA_CP_EBCDIC=IBM-1047
/*
```

Shutting down Process Manager

1. Log in as a superuser, with a UID of 0.
2. Prepare the Process Manager environment. Export the following variables using z/OS UNIX System Services shell commands:

   ```
   export PATH="/usr/lpp/bpa/bin:/usr/lpp/bpa/samples:${PATH}"
   export LIBPATH="/usr/lpp/bpa/lib:${LIBPATH}"
   export NLSPATH="/usr/lpp/bpa/nls/msg/%L/%N:${NLSPATH}"
   ```
3. Enter **p_dae** to execute the shutdown script.

   Note: All server processes controlled by Process Manager will be terminated with the Process Manager shutdown.

Starting the garbage collector

Process Manager has a built-in garbage collector capability to clean-up the managed address spaces and keep internal control information consistent with actual processes running. We recommend you run the garbage collector on a regular basis. Use the **cron** utility to help automate this process.

1. Log in as a superuser, with a UID of 0.
2. Prepare the Process Manager environment. Export at least following variables using z/OS UNIX System Services shell commands:

   ```
   export PATH="/usr/lpp/bpa/bin:/usr/lpp/bpa/samples:${PATH}"
   export LIBPATH="/usr/lpp/bpa/lib:${LIBPATH}"
   export NLSPATH="/usr/lpp/bpa/nls/msg/%L/%N:${NLSPATH}"
   ```
3. Enter **s_gc** to execute the garbage collector start-up script.

End of NOT Programming Interface information

Chapter 3. Application program interface

Connection Manager Application program interface

Connection Manager provides an interface that is similar to open database connectivity (ODBC). Once inside the application program interfaces (APIs), Connection Manager is responsible for presenting the database request to DB2 for z/OS in embedded SQL format. Applications that are already written to use ODBC database APIs are the best suited for using Connection Manager.

This section describes the interfaces that applications use to communicate with Connection Manager. Since Connection Manager is a method for accessing DB2 for z/OS, the applications that use Connection Manager will be database driver type products. Within this section, the terms *application* and *database driver* are synonymous.

This section of the document assumes that you are familiar with standard database terminology and with the ODBC standard. Knowledge of the ODBC implementation by DB2 (the Call Level Interface) and knowledge of DB2 is also very helpful. For further background information, refer to *DB2 SQL Reference* and *DB2 Application Programming and SQL Guide*.

Connection Manager can be used by applications that have the following characteristics:

- Array fetch and array insert are not used.
- Uncommitted or cursor stability read integrity.
- Supported data types are limited to:
  SQL_BINARY
  SQL_CHAR
  SQL_DATE
  SQL_DOUBLE
  SQL_FLOAT
  SQL_INTEGER
  SQL_REAL
  SQL_SMALLINT
  SQL_TIMESTAMP

Refer to the redbook *Locking in DB2 for MVS/ESA™ Environment* for more information about different types of read integrity.

Application changes

This section describes changes that are needed to the application code in order to use Connection Manager. Once these changes are made, the application can invoke Connection Manager APIs and connect to the database. The macro and function interface prototypes can be found in the *cmxcli.h* header files.

Data management

Database applications generally keep track of statements in some form. In ODBC, a statement is called an *hstmt*, which is a term that identifies an SQL string.

In DB2, the term cursor is used to both identify an SQL string and to describe row position in an answer set. Applications that use Connection Manager are expected to keep track of two data area pointers for each *hstmt*. Typically, a database driver has such a data structure. These data area pointers are input and output structured query language data area (SQLDA) pointers. For background on SQLDAs, refer to *DB2 Application Programming and SQL Guide*. The application is also required to track the type of transaction (uncommitted read versus update) for each *hstmt*.

State and session set macro

Since Connection Manager funnels multiple user sessions over a single DB2 connection, state information is required to determine commit and rollback activity, as well as detection of concurrent active updates.

The state information indicates whether the transaction type is:

- an uncommitted read (CM_DTR_TRANS)
- an update type (CM_UPD_TRANS)
- a commit/rollback (CM_COMMIT_RB)

In addition, the logical connection ID is required. This logical connection is equivalent to a connection number for the application. In some cases, applications have only one logical connection; in other cases, applications request multiple logical connections that are referred to here as different sessions. The session_id is expected to be some integer number identifying the application's logical connection. To set the state and session information, invoke the CMX_SET_STATE macro provided by Connection Manager in the *cmxcli.h* header file. For example:

```
CMX_SET_STATE( session_id, transaction_type);
```

The correct place to invoke this macro is at the point when the driver application has accepted a request. Even if multiple SQLs are generated from this one request, the CMX_SET_STATE invocation remains in effect. At each subsequent request, the macro should be invoked to identify the transaction type and session identifier.

Create and fill in SQLDA

The interface to Connection Manager requires the SQLDA to be created and filled in by the database driver. The SQLDAs must exist and must be properly filled in at the time of SQLExecute and SQLFetch. The cmx_alloc_sqlda() function call, provided by Connection Manager, should be used to obtain the SQLDA:

```
struct sqlda *cmx_alloc_sqlda(int num_fields);
```

Where num_fields is the number of SQLVARs that will be used for this SQLDA.

The SQLDAs for SELECT type processing should be obtained at the point where the application binds the input and output parameters to the data fields. Both input and output SQLDAs are required. Once obtained, the SQLDA pointers for an *hstmt* should be tracked by the application, as mentioned in "Data management" on page 25. INSERT, UPDATE, and DELETE processing has only one input SQLDA.

Once obtained, the SQLDA needs to be updated to reflect the sqlvar information. To do this, first invoke the Connection Manager macro that provides addressability to the sqlvar area, based on the number of the variable:

```
CMX_GET_SQLVAR_PTR(sqlvar_ptr,sqlda_ptr,bind_nr);

 where:
    struct sqlvar      *sqlvar_ptr    -> address of sqlvar area
    struct sqlda       *sqlda_ptr     -> sqlda pointer
    int                bind_nr        -> column bind number
```

Once the sqlvar_ptr address is known, invoke the MAKE_SQLVAR macro with the appropriate data values to fill into the SQLDA:

```
CMX_MAKE_SQLVAR(sqlvar_ptr,type,db_size,src_addr,db_name,
            sqlda_ptr,cm_charforbit);

 where:
    struct sqlvar      *sqlvar_ptr    -> address of sqlvar area
    int                type           -> sqltype
    int                db_size        -> sqllen value
    char               *addr          -> sqldata
    char               *name          -> sqlname
    struct sqlda       *sqlda_ptr     -> address of sqlda area
    int                cm_charforbit  -> 1 = bit data field
                                         0 = not bit data field
```

Note that before the CMX_MAKE_SQLVAR macro is invoked, the logic in the driver should determine whether or not this is a bit data field. Bit data implies that DB2 for z/OS will not perform any translation should code pages be different. An example of a bit data field is an index, or hash column. If the field is bit data, set the cm_charforbit variable to 1 on invoking the CMX_MAKE_SQLVAR macro.

The CMX_MAKE_SQLVAR macro must be invoked for each input and each output field.

The CMX_GET_ISQLDA(int hstmt) and CMX_GET_OSQLDA(int hstmt) functions can be used by the application to find the input/output SQLDA pointers after they have been obtained.

The CMX_FREE_SQLDA(struct sqlda *sqldaptr) function can be used to free storage associated with the SQLDA.

Prepare cache interface

Connection Manager contains a prepare cache that assigns free cursors to *hstmts*, as needed. As cursors are freed, they can be reassigned to statements that match the cursor characteristics. This section describes the interfaces to the Connection Manager prepare cache, and the application changes that are necessary to use it.

The cache entry structure is found in cmxcli.h. This entry is filled out and the cmx_chk_cache_hit() Connection Manager function is invoked to determine if a cache entry exists and also for cursor association with an hstmt. The application code that constructs the SQL string and issues an SQLPrepare ODBC request should invoke the cmx_chk_cache_hit() function. The function should be called prior to invoking the SQLPrepare.

The callers of cmx_chk_cache_hit() fill out the input cache entry with the following information:

- table name – reserved, set tbl_name[0] to 0x00
- select type – reserved integer, set to 0
- index name – reserved, set index_name[0] to 0x00
- mode, which is an integer that differentiates the actual SQL being sent to DB2. See file *cmxcli.h* for examples of these Connection Manager constants. It is the logical OR of these flags that determines the locking mode and relational operation of the SQL statement.

The cache entry, hstmt, and the sql statement string are then passed to the cmx_chk_cache_hit() routine:

```
chk_cache_rc = cmx_chk_cache_hit(cache_entry,hstmt,sql_stmt);
```

If the return code is CM_NO_CACHE_HIT, an entry is not found in the Connection Manager cache. If the return code is 0, the entry matches an existing and available cache entry. In the case of a CM_NO_CACHE_HIT, the SQL for the statement must be constructed and a PREPARE issued.

The application's prepare cache logic should resemble the following sequence:

- The application receives a request and uses the CMX_SET_STATE Connection Manager macro to set the state and session identity.
- The application then proceeds to invoke the SQLExecute ODBC call to perform an SQL (without issuing SQLPrepare first).
- If the return code is 0, nothing else is required.
- If DB2 did not find the cursor prepared, a *prepare fault* code is returned. The return code should be checked by calling the CMX_PREPARE_ERR macro. This macro will return a 1 if it was a prepare error; 0 otherwise.
- If it is a *prepare fault* and the cursor was in the Connection Manager cache but cast-out, a CM_CURSOR_CASTOUT return code is received. In this case, the application should invoke cmx_chk_cache_hit(), then issue an SQLPrepare ODBC call.
- If it is a *prepare fault* and the cursor was never in the Connection Manager cache, the CMX_PREPARE_ERR macro returns true, and the application should issue an SQLPrepare without first invoking the cmx_chk_cache_hit() function. This avoids recursive checks of the Connection Manager cache.
- The SQLExecute ODBC call can then be issued, and a SQLFetch request can be issued to get the rows.

Note that in the above case, an SQLPrepare is done only when there is a *prepare fault*. Since SQLPrepare is costly in terms of performance, the Connection Manager cache logic attempts to minimize the prepares.

Commit and rollback

Some changes must be made to the application's commit and rollback logic. When Connection Manager detects that there are multiple updates in progress, and there are multiple commit scopes outstanding, Connection Manager creates an adjunct thread. This means that, for a short amount of time, an application may have more than one physical connection to the database. The term ″thread″ is used to refer to a database thread.

Commit and rollback can occur on the main thread, or the adjunct thread(s), depending on the session and state information. When the application issues an SQLTransact ODBC call to perform commit or rollback, the return code may be one of the following:

0
Commit or rollback issued.

CM_RC_CMRB_NOTISSUED
Commit or rollback not issued because there was no commit scope outstanding.

CM_RC_COMMITRB_ADJUNCT
Commit or rollback issued, but on an adjunct thread due to concurrent multiple updates.

It is important to know whether the commit or rollback was issued on the main or an adjunct thread because of hstmt handling after the SQLTransact call. Following the commit/rollback, the cmx_free_up_cursor() function is invoked for some of the hstmts associated with this logical connection. The application should cycle through all hstmts for the session that is being committed or rolled back. It should determine which hstmts to free, based on the following logic for each *hstmt*:

- If the hstmt is an update type transaction state (part of the data that should be tracked by the application), and the commit/rollback was on a main thread, the cmx_free_up_cursor(hstmt) Connection Manager routine is called; then the hstmt should be freed in the application logic. Note that the return code from SQLTransact indicates whether the commit was on a main or adjunct thread.
- If the hstmt is an update type transaction state, and the commit/rollback was on an adjunct thread, the hstmt should be freed in the application logic. Note that the Connection Manager function cmx_free_up_cursor does not need to be called.
- If the hstmt is an uncommitted read type transaction state, the hstmt does not need to be deleted in the application logic. cmx_free_up_cursor() does not need to be invoked.

The transaction type of each *hstmt* should be tracked by the application in a data structure that represents the *hstmt*. This is mentioned in “Data management” on page 25.

DB2 will mark all cursors for the connection closed automatically upon issuing a commit or rollback. The application must mark all of the hstmts closed after a successful SQLTransact call. Since multiple logical connections for the application are funneled by Connection Manager into one physical connection to DB2, all hstmts for all logical connections need to be marked closed. Note that the cursors are not deleted.

Table-level operations

Since table operations are considered update mode, the CMX_SET_STATE Connection Manager macro is invoked with the appropriate session ID, followed by CM_UPD_TRANS flag. An example is:

```
CMX_SET_STATE( session_id, CM_UPD_TRANS );
```

Build changes

Connection Manager Parts List:

cmxcli.h
Connection Manager contains definitions for the Connection Manager interfaces for functions and macros used. It also contains definitions for variables that are defined. This header file is called cmxcli.h, and it is included as source with Connection Manager. This include file is located in */usr/lpp/cmx/include* after Connection Manager is installed.

cmxdll
The Connection Manager routines and services are packaged into a DLL (dynamic link library) whose path should be made available to the application at runtime by placing it in $LIBPATH. The dll is located in */usr/lpp/cmx/lib/cmxdll*.

cmxdll.x
The Connection Manager routines and service interfaces are packaged into an import file that should be linked with the application code during the application build process. The import file is located in */usr/lpp/cmx/lib/cmxdll.x*.

cmxver
The Connection Manager binary that queries the cmxdll and returns the version and service level. The cmxver executable is located in /usr/lpp/cmx/bin.

’hlq.SCMXDBRM’
: SCMXDBRM is an z/OS partitioned data set that contains DB2 for z/OS pre-compiler output from Connection Manager routines. It is used as input to the BIND process that is required to run applications that use DB2 for z/OS and Connection Manager.

CMXBIND
: This is a sample BIND job that can be used to bind the Connection Manager pre-compiler output to a DB2 for z/OS plan name. Refer to *DB2 Data Sharing: Planning and Administration* for more information on BIND options and plan names. The CMXBIND sample job is a member that is installed into SYS1.SAMPLIB and contains the JCL statements required to BIND.

Required Changes to Application Build:

cmxcli.h
: The cmxcli.h header must be included in every application source file that issues ODBC type calls, as well as any application source that will invoke the Connection Manager interface calls.

C header files
: The DB2 for z/OS C header files are usually placed into an MVS™ partitioned data set during DB2 for z/OS installation. This data set must be specified during the compile step for an application by using the following compile option:

```
-I"//'pds.where.c.headeris.located'"
```

cmxdll.x
: The Connection Manager cmxdll.x file needs to be included on the list of objects for the link during application build.

Restrictions and limitations

There are some restrictions for Connection Manager use:

1. Array insert and array fetch are not emulated because there is marginal, if any, performance benefit.

There are also certain pre-defined Connection Manager limits:

1. A maximum of 10 adjunct threads active at one time for a total of 11 (1 main + 10 adjunct) concurrent commit scopes across all sessions for a single user.
2. A maximum of 500 hstmts allocated on an adjunct thread. Note that an adjunct thread is live only until the next commit.
3. A maximum of 10000 allocated statement handles for 1 active user. This is managed across 1000 available cursors on the main thread and a possible 500 cursors across each of 10 possible adjunct threads.
4. A maximum of 150 date columns in one table row.
5. A maximum of 150 timestamps in one table row.
6. A maximum of 8-character index names.
7. A maximum of 32-character table names.
8. A maximum of 2000 users. DB2 for z/OS allows a maximum of 2000 local connections. As a result, Connection Manager allows a maximum of 2000 users on each z/OS image. Connection Manager allows each user to have multiple logical connections that are handled through a single physical connection to the database.

API

This section describes each of the supported interfaces to Connection Manager. In most cases, there is strong resemblance to ODBC APIs such as DB2 Call-Level Interface implementation. The application maintains the ODBC interface, and Connection Manager intercepts these ODBC calls, performs connection funneling and management, and invokes DB2 for z/OS directly. The remapping of ODBC calls into the Connection Manager calls is done by including the cmxcli.h header file into the application at compile time.

cmx_alloc_sqlda

cmx_alloc_sqlda allocates an SQLDA, which is a 16-byte header plus 44 bytes per field in length. It returns a pointer of type sqlda, as designated in the cmxcli.h header file.

```
struct sqlda *cmx_alloc_sqlda(int num_fields)
```

cmx_check_cursor_castout

cmx_check_cursor_castout determines whether a particular hstmt has been cast out of the CMX prepared statement cache. A response of CM_CURSOR_CASTOUT will be returned in this event. In this case the application is required to acquire a new cursor for the hstmt.

```
int cmx_check_cursor_castout(int hstmt);
```

cmx_chk_cache_hit

cmx_chk_cache_hit determines whether a previously issued prepare and cursor is available for a given hstmt. It tries to match an input cache entry with the Connection Manager manager prepare cache. See the cmxcli.h header file for a description of a CM_PREP_CACHE_ENTRY. It returns CM_NO_CACHE_HIT if not found, 0 otherwise. A DB2 for z/OS cursor will be associated with this hstmt. When not found, the prepare cache is updated with this new entry.

```
int cmx_chk_cache_hit(CM_PREP_CACHE_ENTRY in_cache_entry, int hstmt);
```

cmx_connect_hstmt_isqlda

cmx_connect_hstmt_isqlda associates an input hstmt with a previously allocated input sqlda. The response to the application is an integer return code.

```
int cmx_connect_hstmt_isqlda(int ihstmt, struct sqlda *sqlda_ptr);
```

cmx_connect_hstmt_osqlda

cmx_connect_hstmt_osqlda associates an input hstmt with a previously allocated output sqlda. The response to the application is an integer return code.

```
int cmx_connect_hstmt_osqlda(int ihstmt, struct sqlda *sqlda_ptr);
```

cmx_free_sqlda

cmx_free_sqlda frees a previously allocated sqlda. It takes a pointer of type sqlda as input and returns an integer return code.

```
int  cmx_free_sqlda(struct sqlda *sqlda_ptr);
```

cmx_free_up_cursor

cmx_free_up_cursor disassociates an input hstmt with an actual DB2 for z/OS prepared statement. This makes the statement available for reuse by another hstmt when requested. The response to the application is an integer return code.

```
int cmx_free_up_cursor(int hstmt);
```

cmx_get_isqlda

cmx_get_isqlda returns the input sqlda pointer for an input hstmt.

```
struct sqlda *cmx_get_isqlda(int hstmt);
```

cmx_get_osqlda

cmx_get_osqlda returns the output sqlda pointer for an input hstmt.

```
struct sqlda *cmx_get_osqlda(int hstmt);
```

SQLAllocConnect

This call is disabled by Connection Manager. All connection attributes are established at SQLAllocEnv time. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLAllocConnect( SQLHENV        henv,
                             SQLHDBC FAR *  phdbc );
```

SQLAllocEnv

Application initialization calls SQLAllocEnv to establish the operating environment of the driver. The cmx.conf file is read to determine the actual DB2 for z/OS attach type, code page translation, if any, and the plan name for all subsequent dynamic SQL. At the completion of this call, the primary connection to DB2 for z/OS is established.

```
SQLRETURN __SQLAllocEnv( SQLHENV FAR * phenv );
```

SQLAllocStmt

SQLAllocStmt returns a pointer to a place holder which represents a statement resource. These are equivalent to the hstmt resources generated by the standard ODBC interface.

```
SQLRETURN __SQLAllocStmt( SQLHDBC        hdbc,
                          SQLHSTMT FAR * phstmt)  ;
```

SQLBindCol

This call is disabled by Connection Manager and is replaced by the Connection Manager CMX_MAKE_SQLVAR macro which associates, in an SQLDA entry, an application variable and an output parameter marker. This macro should be invoked in the driver code as outlined in “Application changes” on page 25. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLBindCol( SQLHSTMT         hstmt,
                        SQLUSMALLINT     icol,
                        SQLSMALLINT      fCType,
                        SQLPOINTER       rgbValue,
                        SQLINTEGER       cbValueMax,
                        SQLINTEGER FAR * pcbValue);
```

SQLBindParameter

This call is disabled by Connection Manager and is replaced by the Connection Manager CMX_MAKE_SQLVAR macro which associates an application variable and an input parameter marker in an SQLDA entry. This macro should be invoked in the driver code as outlined in “Application changes” on page 25. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLBindParameter( SQLHSTMT         hstmt,
                              SQLUSMALLINT     ipar,
                              SQLSMALLINT      fParamType,
                              SQLSMALLINT      fCType,
                              SQLSMALLINT      fSqlType,
                              SQLUINTEGER      cbColDef,
                              SQLSMALLINT      ibScale,
                              SQLPOINTER       rgbValue,
                              SQLINTEGER       cbVAlueMax,
                              SQLINTEGER FAR * pcbValue);
```

SQLConnect

This call is disabled by Connection Manager. Authorization checks occur at SQLAllocEnv time during the connection to DB2 for z/OS. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLConnect ( SQLHDBC        hdbc,
                         SQLCHAR FAR *   szDSN,
                         SQLSMALLINT     cbDSN,
                         SQLCHAR FAR *   szUID,
                         SQLSMALLINT     cbUID,
                         SQLCHAR FAR *   szAuthString,
                         SQLSMALLINT     cbAuthString );
```

SQLDisconnect

This call is disabled by Connection Manager. A connection to the database is made at SQLAllocEnv time. This main thread is maintained until what is known at end-of-task (EOT) time. For adjunct threads, which only persist while multiple commit scopes are active, an implicit disconnect is done at thread termination. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLDisconnect( SQLHDBC hdbc );
```

SQLExecDirect

SQLExecDirect is used for immediate execution of an SQL string when no input is required. Typically, this is DDL (data definition language). This is not a performance path, so a full prepare and execute is done for all SQLExecDirect calls.

```
SQLRETURN __SQLExecDirect( SQLHSTMT      hstmt,
                           SQLCHAR FAR * SzSqlStr,
                           SQLINTEGER    cbSqlStr);
```

SQLExecute

SQLExecute is similar in function to the ODBC equivalent.

```
SQLRETURN __SQLExecute( SQLHSTMT hstmt);
```

SQLExtendedFetch

This call is disabled by Connection Manager. Array fetch is not supported in DB2 for z/OS. Each row must be individually fetched. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLExtendedFetch( SQLHSTMT           hstmt,
                              SQLUSMALLINT       fFetchType,
                              SQLINTEGER         irow,
                              SQLUINTEGER FAR *  pcrow,
                              SQLUSMALLINT FAR * rgfRowStatus);
```

SQLFetch

SQLFetch is similar in function to the ODBC equivalent.

```
SQLRETURN __SQLFetch( SQLHSTMT hstmt);
```

SQLFreeConnect

This call is disabled by Connection Manager. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLFreeConnect( SQLHDBC hdbc );
```

SQLFreeEnv

This call closes and flushes the trace file when tracing is active. All other environment clean-up is done implicitly at end-of-task (process end) time.

```
SQLRETURN __SQLFreeEnv( SQLHENV henv );
```

SQLFreeStmt

SQLFreeStmt fOptions (except for SQLDROP) are disabled by Connection Manager. If a cursor is open, this option closes the DB2 for z/OS cursor. It frees the hstmt and marks the DB2 for z/OS statement ″available″ in the Connection Manager prepare cache.

```
SQLRETURN __SQLFreeStmt( SQLHSTMT     hstmt,
                         SQLUSMALLINT fOption);
```

SQLParamOptions

This call is disabled by Connection Manager. SQLParamOptions are used only for setting the array size to be used during array fetch operations and is not supported by Connection Manager. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLParamOptions( SQLHSTMT            hstmt,
                             SQLUINTEGER         row_arrsz,
                             SQLUINTEGER FAR  *   rpc);
```

SQLPrepare

SQLPrepare is similar in function to the ODBC equivalent interface. See “Prepare cache interface” on page 27.

```
SQLRETURN __SQLPrepare( SQLHSTMT      hstmt,
                        SQLCHAR FAR * szSqlStr,
                        SQLINTEGER    cpSqlStr);
```

SQLSetConnectOption

This call is disabled by Connection Manager. The Autocommit feature is disabled at BIND time of the SQL plan. Transaction isolation is established as needed on individual SQL statements for DB2 on z/OS. For example, cursor stability (COMMITTED READ) can be established for a read by appending WITH CS to a SELECT type SQL statement to satisfy locking requirements. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLSetConnectOption( SQLHDBC        hdbc,
                                 SQLUSMALLINT   fOption,
                                 SQLUINTEGER    vParam );
```

SQLSetEnvAttr

This call is disabled by Connection Manager. SQL is passed as a variable length string to DB2 for z/OS with a 2-byte length prefix. This is used instead of null termination. The MAXCONN environment attribute is overridden by the connection funneling characteristics of the driver. Whenever there is no more than one commit scope outstanding, only one actual DB2 for z/OS connection is established per driver. When multiple commit scopes are active (a rare occurrence), a separate connection is established for each. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLSetEnvAttr( SQLHENV       henv,
                           SQLINTEGER    Attribute,
                           SQLPOINTER    Value,
                           SQLINTEGER    StringLength );
```

SQLSetStmtOption

This call is disabled by Connection Manager. Certain statement-level options are set at when the DB2 for z/OS plan is bound to the application. For example, CURSOR(NOHOLD) is set globally for Connection Manager when the DB2 for z/OS plan is created. The BIND is done statically as a customization step prior to execution of the application. Statement-level isolation is done as required for locking purposes by appending the correct locking clause (for example, WITH RS for read stability) to the SQL string. The API is left for completeness and to minimize source changes.

```
SQLRETURN __SQLSetStmtOption( SQLHSTMT     hstmt,
                              SQLUSMALLINT type,
                              SQLUINTEGER  param);
```

SQLTransact

SQLTransact is similar in operation to that of the ODBC interface. Connection Manager uses the session ID and state information to determine whether a commit or rollback needs to be done. An action (COMMIT or ROLLBACK) is required only if an update (DDL, insert, update, delete) has been done to the database in the current commit scope. If commit or rollback needs to be done, Connection Manager determines whether it should be done on the main or an adjunct thread.

```
SQLRETURN __SQLTransact( SQLHENV      henv,
                         SQLHDBC      hdbc,
                         SQLUSMALLINT fType);
```

Process Manager Application program interface

Process Manager is a unique IBM interface. If you are interested in obtaining more information about Process Manager, please contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY, USA 10504-1785

Chapter 4. Problem determination and messages

Connection Manager Problem determination and messages

This section explains the problem determination tools available in Connection Manager and lists all Connection Manager messages.

Connection Manager Problem determination

This section explains the problem determination tools available in Connection Manager.

Error logging

The Connection Manager writes error messages to a log file in the HFS. The CMX_ERROR_LOG environment variable specifies a path and filename to be used as the log. If not specified, the file *cmx.log* will be created in the */tmp* directory. An example of setting this environment variable is:

```
export CMX_ERROR_LOG=/tmp/cmx/cmx.log2
```

Connection Manager messages are prefixed with CMX. See "Connection Manager Messages" on page 40.

User tracing

In addition to the error message logging, you can optionally trace activity for a single user, list of users, or all users. Each user traced will produce a trace file. The CMX_TRACE_DIR environment variable can be used to specify the directory where traces will be placed. The file name for each trace is *cmxtrace.username* (where username is different for each user). If this environment variable is not set, the trace file will reside in */tmp*. An example of setting this environment variable is:

```
export CMX_TRACE_DIR=/tmp/cmx/trace
```

Depending on the settings in the */etc/cmx/cmx.conf* configuration file, any existing trace file will either be appended or replaced. To turn this trace on, set Trace=1 in the TRACE section of the */etc/cmx/cmx.conf* configuration file. You can specify a maximum line count for each trace file by setting the TraceWrap value. Note this is the line count, not byte count. Once the file reaches the stated linecount, it begins to wrap, so at any given time you have the last trace entries recorded.

If TraceWrap is set to greater than 0, any existing file with the same name will get replaced. If TraceWrap is 0, trace will be appended to an existing file, or create one if no file exists. This TraceWrap is provided to avoid using space in */tmp* unnecessarily. If you run with TraceWrap=0 and continuously trace a set of users, you should carefully monitor your file system to make sure you have enough space. The trace values are read once only when initializing each user, so changes to the configuration file are not reflected dynamically.

The third parameter in the TRACE section of */etc/cmx/cmx.conf* file is the TraceUser option. For this option, you may specify:

one_username
The userid of a single user.

ALL The keyword ALL.

path_and_filename
The full path and filename of a file that contains a list of users.

© Copyright IBM Corp. 1998, 2002

If Trace=1 (tracing is on), you **must** specify something for the TraceUser option. There is no default for this option. If you specify a path and filename, the file should contain a list of users, one per line. For example:

```
TraceUser=/tmp/cmx/user.file

user.file contains:
       JOE
       MARY
       CHRIS
```

The fourth parameter in the TRACE section of the /etc/cmx/cmx.conf file is the TraceShort option. For this option you may specify a value of 1 or 0 (the default). For example:

```
TraceShort=1
```

This specification will cause the SQL statement traced on a PREPARE statement to be truncated at 136 bytes. This might be needed in cases where the SQL statements are exceedingly long and would easily fill up a trace log. This option is ignored under Level 1 SQL because the SQL statement is not traced.

The final parameter in the TRACE section of */etc/cmx.conf* file is the TraceTables option. For this option, you may specify a detailed trace directed at specific tables in a list. For example:

```
TraceTables=Table1,Table2,Table3
```

This specification will cause, in addition to normal trace output, a formatted dump of both input and output SQLDA for tables Table1, Table2, and Table3.

Trace format

The format of the trace output file is:

- The first trace entry has a timestamp that the user was connected to the database.
- The second trace entry has the subsystem, planname, trace=on or off, trace wrap amount, the number of tables in detailed trace, whether all SQL errors will be logged in log.cm, and which SQL errors to ignore. This is essentially all the */etc/cmx/cmx.conf* settings.

The following records are the summary trace of actual execution:

```
date time: stmttype session hstmt cursor mode indexname tablename sqlstatement
```

date/time
Timestamp of trace entry to millisecond accuracy.

stmt type
Statement type, which can be PREP, OPEN, FTCH, ISRT, DELT, UPDT, CLOS, CMIT, ROLB, EXDR, EXEC, CADD, CLRU, LRUO, FRCR, CMIT MAIN (actual commit on main thread), or ROLB MAIN (actual rollback on main thread). Any ″type″ postpended with ″THD″ means the action was done on an adjunct thread.

session
Session number or logical connection number

hstmt hstmt value for this statement

cursor
The CM cursor used to represent this hstmt. Note that the hstmt and cursor do not have to be the same since the CM reuses cursors.

mode The mode of the statement that can be interpreted as follows:

- If stmttype is ISRT,DELT,FTCH, PREP, CLOS represents the corresponding action
- If stmttype is OPEN, and if the low-order bit is on in the mode, then the stmt is a SELECT with LOCK.
- If stmttype is OPEN, and if the low-order bit is not on in the mode, then the stmt is a SELECT without LOCK.

indexname
The index name used. SQL Level 1 only.

tablename
The tablename. SQL Level 1 only.

sqlstatement
The SQL statement being processed. SQL Level 2 only.

Any trace that pertains to adjunct threads contains the information THD=xx, where xx is the thread number. You should see very few of these entries, because multiple concurrent update scopes are an infrequent occurrence.

Detailed table trace

The following is the SQLDA dump format for both an input (OPEN) SQLDA and output (FETCH) SQLDA, which includes:

- The SQLDA address and header information
- The SQLVAR information, with one line for each SQLDA variable
  - the variable number, starting from 0
  - the SQLTYPE (CHAR, INT, SHORT INT, etc.). The ODBC integer types are listed, in integer format, in the *sql.h* ODBC header file.
  - the SQLLEN, the length of the variable
  - a marker, which when set to 08 implies use of a CCSID for a character variable
  - the CCSID of a character variable when enabled, FFFF when character for bit data, or 0.
  - 30 bytes of input or output data associated with the variable

Here is an example of the detail for TraceTables=tttaad100000:

```
98-10-06[11:30:04.238]:OPEN 0001 0007 0007 00128 hash1 empower.tttaad100000
BEGIN SQLDA
sqlda: 0e9c5b48 sqldaid= SQLDA+I sqldabc=0104 sqln=0002 sqld=0001

0000 452 03 08 FFFF 805102
END SQLDA

98-10-06[11:30:04.245]:FTCH 0001 0007 0007 00128 hash1 empower.tttaad1000

BEGIN SQLDA
sqlda: 0e9c5bb8 sqldaid= SQLDA+O sqldabc=0456 sqln=0010 sqld=0010

0000 500 02 00 0000 812
0001 452 30 08 0333 42656E63686D61726B206461746120202020202020202020202
0002 452 01 08 FFFF 01
0003 452 03 08 0333 4E4C47
0004 452 08 08 0333 424D5F3030312020
0005 452 08 08 0333 424D5F3030312020
0006 496 04 00 0000 0
```

```
0007 496 04 00 0000 0
0008 452 03 08 FFFF 805102
0009 452 11 08 FFFF 424D5F3030312020805102
END SQLDA
```

Problem determination environment variables

There are a set of Connection Manager environment variables that are useful in problem determination:

```
CMX_TRACE_SEP
 Description:
    The CMX_TRACE_SEP environment variable can be used
    to produce separate trace files when running multiple users
    with the same userid.  As a result of setting this environment
    variable, Connection Manager will append a PID (process id)
    to the name of the trace file.
    export CMX_TRACE_SEP=1
 Default:
    0

 CMX_TRACE_DEBUG
 Description:
    The CMX_TRACE_DEBUG environment variable forces a trace
    file flush after each trace update.  This can be extremely
    useful when diagnosing an abnormal end.
 Example:
    export CMX_TRACE_DEBUG=1
 Default:
    0

CMX_TRACE_SPECIAL
 Description:
     This environment variable specifies a DB2 error sqlcode.
     If this is set and the sqlcode is encountered, Connection
     Manager will print the input or output sqlda to the
     cmxtrace file.  This environment variable can be used
     in conjunction with the existing Connection Manager trace
     facility.
 Example:
     (will print the sqlda to the cmxtrace file if
      sqlcode -302 is encountered)
     export CMX_TRACE_SPECIAL=-302
 Default:
     Not set
```

Connection Manager Messages

Connection Manager messages are logged into the error log file. You can specify the path and filename with the **CMX_ERROR_LOG** environment variable. If this variable is not specified, */tmp/cmx.log* file is produced.

The syntax of each line of the error log is as follows:

```
CMXnnnE date time user file(line#) function: return codes  text
```

Where:

CMXnnnE
Message number

date Date the error occurred

time Time the error occurred

file(line#)
C-file and line number reporting the error

function
C-function where the error was detected

text Message text and supplemental error information

Each error description contains the following fields:

Explanation
Describes the reason of this error message in detail level

User Response
What you should do in response

System Programmer Response
What the system programmer (the Connection Manager Administrator) should do in response

General Remarks:

- Some errors are Connection Manager specific and not related to DB2 for z/OS. For example, if the Connection Manager configuration file cannot be read, or if there are invalid or inconsistent parameters specified, errors will be produced and logged. Other examples of Connection Manager errors include: DB2 for z/OS modules cannot be loaded, Connection Manager cursor or statement management is inconsistent, internal message buffer is too small, or no statements are available to allocate.

  Some errors reported by Connection Manager reflect the sqlcode produced by DB2 for z/OS in response to SQL. These sqlcodes can be found in the *DB2 Messages and Codes*. Application errors may cause DB2 for z/OS sqlcodes.
- Connection Manager errors have the format CMXnnnE, where nnn ranges currently from 1-100. There are ranges that indicate the type of error:

  CMX001E-CMX025E: Connection Manager initialization failures

  CMX026E-CMX050E: Connection Manager internal errors not related to DB2 for z/OS

  CMX051E-CMX075E: Connection Manager report of negative sqlcodes received from DB2 for z/OS

  CMX076E-CMX100E: Connection Manager adjunct thread errors.
- Connection Manager provides a message catalog (cmxerror.cat) that is placed in the HFS by the SMP/E installation.

CMX001E Resource Read failure on config file

Explanation: This message should be accompanied by a CMX002E - CMX007E message indicating the cause of the configuration file read failure.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the accompanying CMX002E - CMX007E and correct the reported configuration file failure.

CMX002E Resource config file cmx.conf is not valid

Explanation: The default configuration file could not be opened.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: The default configuration file is */etc/cmx/cmx.conf.* Make sure this file exists and has correct access permissions for users.

CMX003E Resource config file specified by CMXCONF environment variable not valid

Explanation: The CMXCONF environment variable is set, but the configuration file pointed to by the CMXCONF environment variable could not be opened.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check that the CMXCONF environment variable contains a valid path and file name and that the access permissions for users are correct. Note that if the CMXCONF variable is not set, the default configuration file */etc/cmx/cmx.conf* is read.

CMX004E DB2SubSystem in config file is not valid

Explanation: The DB2SubSystem parameter in the configuration file is either not present, not in the NAMES section, or is greater than 4 characters.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check that the DB2SubSystem parameter is specified, that it is found in the NAMES section of the configuration file, and that it is only four characters in length. Also ensure that the configuration file being read is the desired one.

CMX005E DB2PlanName in config file is not valid

Explanation: The DB2PlanName parameter in the configuration file is either not present, not in the NAMES section, or is greater than eight characters.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check that the DB2PlanName parameter is specified, that it is found in the NAMES section of the configuration file, and that it is only eight characters in length. Also ensure that the configuration file being read is the desired one.

CMX006E DB2Attach type in config file is not valid -- RRS will be used

Explanation: The DB2Attach parameter in the configuration file is not set to either *CAF* or *RRS*.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check if a DB2Attach parameter is included in the NAMES section of the configuration file. If present, verify that it is set to either *CAF* or *RRS*; the default is RRS. Also ensure that the configuration file being read is the desired one.

CMX007E TraceUser in config file is not valid

Explanation: In the TRACE section of the configuration file, CMTrace is set to 1, but CMTraceUser is not specified, or CMTraceUser exceeds a maximum file path length of 60.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the CMTrace and CMTraceUser parameters in the configuration file and ensure that CMTraceUser is set if CMTrace is 1.

CMX008E Failure loading DSNTIAR

Explanation: DB2 module DSNTIAR cannot be loaded.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the z/OS search order the DB2 SDSNLOAD library and ensure that users can access the PDS. Also check the z/OS system log for z/OS LOAD failure messages. Check either the STEPLIB, LNKLST or LPA concatenations.

CMX009E Failure loading DSNALI

Explanation: DB2 module DSNALI cannot be loaded.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the z/OS search order the DB2 SDSNLOAD library and ensure that users can access the PDS. Also check the z/OS system log for z/OS LOAD failure messages. Check either the STEPLIB, LNKLST or LPA concatenations.

CMX010E Failure loading DSNHLI2

Explanation: DB2 module DSNHLI2 cannot be loaded.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the z/OS search order the DB2 SDSNLOAD library and ensure that users can access the PDS. Also check the z/OS system log for z/OS LOAD failure messages. Check either the STEPLIB, LNKLST or LPA concatenations.

CMX011E Failure loading DSNRLI

Explanation: DB2 module DSNRLI cannot be loaded.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the z/OS search order the DB2 SDSNLOAD library and ensure that users can access the PDS. Also check the z/OS system log for z/OS LOAD failure messages. Check either the STEPLIB, LNKLST or LPA concatenations.

CMX012E Failure loading DSNHLIR

Explanation: DB2 module DSNHLIR cannot be loaded.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Check the z/OS search order the DB2 SDSNLOAD library and ensure that users can access the PDS. Also check the z/OS system log for z/OS LOAD failure messages. Check either the STEPLIB, LNKLST or LPA concatenations.

CMX013E DB2 CAF CONNECT failure

Explanation: Failure during a CAF CONNECT to the DB2 subsystem found in the message.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Look up the hexadecimal reason code specified in the message in the *DB2 Messages and Codes*. This will indicate the cause of the CONNECT failure for the subsystem listed in the message text.

CMX014E DB2 CAF OPEN failure

Explanation: Failure during a CAF OPEN to the DB2 subsystem found in the message.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Look up the hexadecimal reason code specified in the message in the *DB2 Messages and Codes*. This will indicate the cause of the OPEN failure for the subsystem listed in the message text.

CMX015E DB2 RRS IDENTIFY failure

Explanation: Failure during a RRS IDENTIFY to the DB2 subsystem found in the message.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Look up the hexadecimal reason code specified in the message in the *DB2 Messages and Codes*. This will indicate the cause of the IDENTIFY failure for the subsystem listed in the message text. Also check that the RRS subsystem is configured and operational. The z/OS system log can also be checked for RRS-related messages.

CMX016E DB2 RRS SIGNON failure

Explanation: Failure during a RRS SIGNON to the DB2 subsystem found in the message.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Look up the hexadecimal reason code specified in the message in the *DB2 Messages and Codes*. This will indicate the cause of the SIGNON failure for the subsystem listed in the message text. Also check that the RRS subsystem is configured and operational. The z/OS system log can also be checked for RRS-related messages.

CMX017E DB2 RRS CREATE THREAD failure

Explanation: Failure during a RRS CREATE THREAD using the DB2 plan name found in the message.

User Response: Contact your system programmer and report the configuration problem.

System Programmer Response: Look up the hexadecimal reason code specified in the message in the *DB2 Messages and Codes*. This will indicate the cause of the CREATE THREAD failure for the plan name listed in the message text.

CMX026E Cursor out of range

Explanation: The cursor is not in the range from 1 to 2001.

User Response: Contact your system programmer and report the error.

System Programmer Response: Call IBM support with this message and report the error message number and its contents.

CMX027E Invalid statement length

Explanation: A PREPARE request was issued and the string length was greater than 32765.

User Response: The application has issued a PREPARE database request, but the size of the SQL string exceeds 32765. Verify the application's PREPARE request, and ensure that the string length is less than 32765.

System Programmer Response: None.

CMX028E Error message buffer too small

Explanation: An error occurred on a request to DB2, Connection Manager attempted to issue a DSNTIAR call to obtain the full DB2 error text, but error text exceeded the allowed length of 800.

User Response: The error is logged in the Connection Manager error log if the DB2SqlError flag is on in the configuration file. Check this log for the sqlcode associated with this error. Also report this error to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error message number and its contents.

CMX029E DB2 DSNTIAR internal error

Explanation: Connection Manager attempted to issue a DSNTIAR call to DB2. However, DSNTIAR returned a nonzero return code.

User Response: Contact your system programmer and report the failure.

System Programmer Response: Call IBM support with this message and report the error message number and its contents.

CMX051E No available HSTMTs to allocate

Explanation: The number of statements concurrently allocated by the application exceeds the Connection Manager limit of 10000.

User Response: Analyze the application to determine the usage of hstmts, and whether there are errors in statement release logic.

System Programmer Response: None.

CMX052E PREPARE failure during execute direct

Explanation: There was a DB2 error while executing a PREPARE SQL statement during a SQLExecDirect request.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the SQL string being PREPARE'd, along with the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the PREPARE error.

System Programmer Response: None.

CMX053E OPEN failure during execute direct

Explanation: There was a DB2 error while executing an OPEN SQL statement during an SQLExecDirect request for a SELECT type statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*, GC26-9940. The error is for hstmt specified by h= in the message. This message will be followed by the SQL string given as input to SQLExecDirect, and the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the OPEN error.

System Programmer Response: None.

CMX054E EXEC DIRECT failure

Explanation: There was a DB2 error while executing an EXECUTE SQL statement during a SQLExecDirect request for an INSERT,UPDATE, or DELETE statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the SQL string given as input to SQLExecDirect, and the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the EXECUTE error.

System Programmer Response: None.

CMX055E PREPARE failure

Explanation: There was a DB2 error during execution of a PREPARE SQL statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the PREPARE error.

System Programmer Response: None.

CMX056E OPEN failure during execute

Explanation: There was a DB2 error while executing an OPEN SQL statement during an SQLExecute request for a SELECT type statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the OPEN error.

System Programmer Response: None.

CMX057E Failure during INSERT, UPDATE, or DELETE

Explanation: There was a DB2 error while executing an EXECUTE SQL statement request for an INSERT,UPDATE, or DELETE statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. The type of statement (ISRT, DELT, UPDT) is specified in the message following the username. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the EXECUTE error.

System Programmer Response: None.

CMX058E FETCH failure

Explanation: There was a DB2 error while executing a FETCH SQL statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the FETCH error.

System Programmer Response: None.

CMX059E CLOSE failure

Explanation: There was a DB2 error while executing a CLOSE SQL statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. The error is for hstmt specified by h= in the message. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the CLOSE error.

System Programmer Response: None.

CMX060E COMMIT failure

Explanation: There was a DB2 error while executing a COMMIT SQL statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the COMMIT error.

System Programmer Response: None.

CMX061E ROLLBACK failure

Explanation: There was a DB2 error while executing a ROLLBACK SQL statement.

User Response: The sqlcode is found in the rc= part of the error message. Find the sqlcode in the *DB2 Messages and Codes*. This message will be followed by the output of the DSNTIAR call to DB2. Analyze the application logic to determine the cause of the ROLLBACK error.

System Programmer Response: None.

CMX076E Pthread signal thread error

Explanation: An error has occurred during the signalling of a database thread.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message. Report any additional CMX077E-CMX088E messages which may accompany this general signalling error.

CMX077E Maximum number of adjunct threads has been reached

Explanation: The internal number of concurrent update DB2 threads has been reached.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX078E Pthread mutex init error

Explanation: The pthread_mutex_init function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX079E Pthread cond init error

Explanation: The pthread_cond_init function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX080E Pthread mutex lock error

Explanation: The pthread_mutex_lock function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX081E Pthread create error

Explanation: The pthread_create function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX082E Pthread cond wait error

Explanation: The pthread_cond_wait function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX083E Pthread mutex unlock error

Explanation: The pthread_mutex_unlock function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX084E Pthread cond signal error

Explanation: The pthread_cond_signal function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX085E Pthread join error

Explanation: The pthread_join function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX086E Pthread detach error

Explanation: The pthread_join function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX087E Pthread cond destroy error

Explanation: The pthread_cond_destroy function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX088E Pthread mutex destroy error

Explanation: The pthread_mutex_destroy function call has failed.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the error number, which immediately precedes the text of the full message.

CMX089E Adjunct thread request invalid

Explanation: The request type passed to an adjunct thread was incorrect.

User Response: Retain this message and show it to your systems programmer.

System Programmer Response: Call IBM support with this message and report the invalid request type, which immediately precedes the text of the full message.

Process Manager Problem determination and messages

This section describes the Process Manager problem determination tools and messages.

Process Manager problem determination

This section describes the problem determination tools available for the Process Manager.

Error logging

The Process Manager writes error messages to a log file in the HFS. The variable *ErrorLog* within group *[BASE]* in the Process Manager configuration file specifies a path and file name to be used as the log file. If this *ErrorLog* variable is not specified, the Process Manager writes an error message to *stderr* and the Process Manager will not start up.

Error log configuration:

Table 9. Errorlog parameter. This entry is used to configure the Error Log.

Req	Entry	Range	Description
Yes	Errorlog	valid absolute pathname	Errorlog determines the pathname of the error log file used by the Process Manager.

Error log format: The format of a Process Manager errors in this log is as follows:

1. Prefix part:
   - BPA (product prefix) (see “Process Manager messages” on page 50)
   - 4 digit error number
   - severity level

     S - severe error

     E - error, medium severe error

     W - warning
   - Time stamp
   - Process ID
   - Thread ID
   - Address space identifier (AS_ID), in hex
   - File name
   - Function name
   - Line number where the error occurred
2. Description string:
   - Error label (eg. RC_RTS_MALLOC)
   - Description of the error

Error string example:

```
BPA0308W  Jun 5 14:40:32 1997 1241513994:00:0x00a2 bpassp.c(00226) main:
          RC_SSP_NOT_IN_SAME_AS This SSP is started outside parent AS
BPA0209E  Jun 5 14:40:32 1997 1241513994:00:0x00a2 bpasspm.c(00247) main:
          RC_SSPM_OPEN_SOCKET_ERROR
BPA2047E  Jun 5 14:40:32 1997 1241513994:00:0x00a2 bpasspm.c(00248) main:
          RC_RTS_SOCKET, EDC5000I No error occurred.  errno2(00000000)
```

Trace facility

In addition to the error message logging, the activities of the Process Manager can be traced. Tracing is based on statements at strategic places within the source code.

Tracing requires an additional process, the trace log daemon, to be started prior to the Process Manager processes. The log daemon is configured during start-up with the same configuration file as the Process Manager *(/etc/bpa/bpares.cfg*). See Table 5 on page 20 for the list that can be set within the configuration file to configure the trace log daemon and the trace behavior. If the number of traces exceeds the number of configured trace records, the trace file will wrap. This will overwrite existing trace strings and will place a marker, >>> *wrap* <<<, in the trace log file.

Trace string: The trace string consists of:
1. Running number (enables sorting after extracting a thread trace)
2. Process ID
3. Thread ID
4. Address space identifier (AS_ID) in hex
5. File name
6. Line number
7. Function name
8. User string

The following information is an example of trace strings:

```
0000001 Jun 02 17:05:03 1997 0101233456:00:0x001a bpassp.c(45) main() SSP thread entered
0000002 Jun 02 17:05:04 1997 0101233456:00:0x001a bpasunxt.c(56) openSocket()
```

Process Manager messages

This section describes the Process Manager error messages that are written to the error log file. Each error description contains the following fields:

Topic **Description**

BPAxxxx
The error message number

RC_<Module>_sssss
The error alias

Explanation
Describes the reason of the error message.

Severity
Describes the severity of this problem:

Warning
In general, those error messages will not affect the function of Process Manager.

Error The function with the error will normally terminate. Depending on the case, the upper-level functions might recover or Process Manager will be terminated with a follow-on error.

Severe
This error will be logged on the system console. Process Manager is not able to perform its work and is stopped. Human intervention is required.

Module
This is the description of the internal module that caused the problem. This information will be helpful for the support team to localize and fix the problem. The following sources will be distinguished:

DAE	Process Manager Daemon main threads
SSPM	server shell process manager
SSP	server shell process
SSPT	server shell process table ADT
SXT	z/OS UNIX Domain Socket
TRC	Messaging and tracing
LOGD	Log Daemon
FUX	Faked z/OS UNIX system calls
SHM	Shared memory handling ADT
GARC	garbage collector
RTS	Run-Time system (C-Run-time services)

System Action
How the component (source) is handling the problem.

System Programmer Response
What the system programmer (the Process Manager Administrator) should do in response.

General Remarks

- All errors are written to the Process Manager error log file. The location of this log file within the HFS file system is defined within the Process Manager configuration file. It is defined within the **BASE** group in the parameter **ErrorLog**.
- **Never** react to problems by issuing commands on the system console. If, for example, there is an error displayed on the system console, the system programmer cannot set any commands (for example, p_dae) from the system console. Instead you must use the rlogin session which was used to start Process Manager.
- SIGINT errors on operator terminal. When terminating Process Manager, some SIGINT messages may appear on the login shell. These messages cannot be suppressed and should therefore be ignored.

BPA0101 RC_DAE_CLOSE_SOCKET

Explanation: Closing an z/OS UNIX domain socket failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0105 RC_DAE_INVALID_MESSAGE_LENGTH

Explanation: Internal error. The total length of a Process Manager internal message is incorrect.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0106 RC_DAE_MESSAGE_HEADER_LENGTH

Explanation: Internal error. The Process Manager header length is incorrect.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0107 RC_DAE_MESSAGE_UNSUPPORTED

Explanation: Internal error. Process Manager received an unsupported internal message.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0109 RC_DAE_OPEN_SOCKET

Explanation: Open of an z/OS UNIX domain socket failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0112 RC_DAE_PASS_SOCKET_WITH_DATA

Explanation: Passing a socket within Process Manager failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0113 RC_DAE_RECEIVEREXEC

Explanation: Receiving the Rexec message failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0114 RC_DAE_UPDATE_SSP_ENTRY

Explanation: Update of a SSP entry in the SSPT failed.

Severity: Error

Module: DAE

System Action: Process Manager might have an inconsistent view about it's controlled processes.

System Programmer Response: Start the garbage collection function. Check the previous error return code(s) to obtain the reason of the error.

BPA0115 RC_DAE_ILLEGAL_ARGC_VALUE

Explanation: Illegal value of argc. The Process Manager daemon is expected to be called with one argument (in the path-name of it's configuration file).

Severity: Error

Module: DAE

System Action: The start-up of the Process Manager daemon fails.

System Programmer Response: Check that Process Manager is called with the path name of it's configuration file.

BPA0116 RC_DAE_GET_SERVER

Explanation: Process Manager cannot supply a client with free server process for an incoming ″REXEC″ request.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0117 RC_DAE_CONFIG

Explanation: Process Manager was started with an unrecoverable bad configuration within it's configuration file.

Severity: Severe

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error messages of the thread in error and fix the configuration.

BPA0118 RC_DAE_TERMSSPTABLEACCESS

Explanation: Termination of the SSPT (server shell process table) failed. The clean-up of processes controlled by the Process Manager might have failed.

Severity: Severe

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0119 RC_DAE_BLOCKSIGNAL

Explanation: A signal could not be blocked.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0121 RC_DAE_INVALID_SSP_ASMT_READY_FOR_WORK_MSG

Explanation: Internal Error. An internal message of Process Manager is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0122 RC_DAE_INITRESOURCEADT

Explanation: Initialization of the vital configuration ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager cannot proceed with start-up.

System Programmer Response: Check that the configuration file passed with Process Manager start-up exists and has the proper access-rights. Check the previous error return code(s) to obtain the reason of the error.

BPA0126 RC_DAE_UNEXPECTED_CLOSED_CONNECTION

Explanation: A connection to Process Manager was closed unexpectedly by a client.

Severity: Error

Module: DAE

System Action: The thread cannot proceed and shuts down. The request is terminated by the client.

System Programmer Response: Check your TCP/IP network. Check the previous error return code(s) to obtain the reason of the error.

BPA0127 RC_DAE_INITSSPTABLEACCESS

Explanation: Initialization of the SSPT ADT failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0130 RC_DAE_INSERT_SSP_ENTRY

Explanation: Inserting a new SSP entry into SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in an inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function.

BPA0131 RC_DAE_INVALID_LCS_ASMT_EXEC_MSG

Explanation: Internal Error. An internal Process Manager message is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0132 RC_DAE_INVALID_LCS_ASMT_EXEC_FAILED_MSG

Explanation: Internal Error. An internal Process Manager message is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0134 RC_DAE_INVALID_LCS_ASMT_EXIT_MSG

Explanation: Internal Error. An internal Process Manager message is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0135 RC_DAE_INVALID_LCS_ASMT_SPAWN_MSG

Explanation: Internal Error. An internal Process Manager message is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0136 RC_DAE_UPDATE_LCS_ENTRY

Explanation: Updating an LCS entry of SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in a inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function

---

BPA0138 RC_DAE_INSERT_AND_SET_LCS_ENTRY

Explanation: Insert and preset of a new LCS entry of SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in a inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function

---

BPA0139 RC_DAE_DELETE_LCS_ENTRY

Explanation: Deletion of an existing LCS entry of SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in a inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function.

BPA0140 RC_DAE_INVALID_SSP_ASMT_EXEC_MSG

Explanation: Internal error. An internal Process Manager message is invalid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0143 RC_DAE_SETSIGCHLDACTION

Explanation: Installation of an action for SIGCHLD failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0144 RC_DAE_SETSIGINTACTION

Explanation: Installation of an action for SIGINT failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0145 RC_DAE_SSPTTERMINATED

Explanation: SSPT has been unexpectedly terminated.

Severity: Warning

Module: DAE

System Action: Process Manager is forced to shut down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0146 RC_DAE_UNBLOCKSIGNAL

Explanation: Unblocking a Signal failed.

Severity: Severe

Module: DAE

System Action: The executing thread shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0149 RC_DAE_MISSING_CONFIGURATION_ENTRY

Explanation: A required configuration parameter entry is missing. See group and parameter name supplied with this message.

Severity: Error

Module: DAE

System Action: Process Manager will shutdown.

System Programmer Response: Check the configuration file used with the start-up of the Process Manager daemon.

BPA0150 RC_DAE_INVALID_CONFIGURATION_ENTRY

Explanation: An incorrect configuration parameter entry has been found. See group and parameter name supplied with this message.

Severity: Warning

Module: DAE

System Action: Process Manager uses the default value for the parameter.

System Programmer Response: Check the configuration file used with the start-up of the Process Manager daemon.

BPA0151 RC_DAE_BUFFER_TOO_SHORT

Explanation: Internal error. Given buffer to short.

Severity: Error

Module: DAE

System Action: The thread cannot proceed and shuts down. The request is terminated by the client.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0152 RC_DAE_TWO_PORT_PROTOCOL_UNSUPPORTED

Explanation: The 2-port protocol of Rexec is not support by Process Manager.

Severity: Error

Module: DAE

System Action: The client request is refused.

System Programmer Response: Use the client with 1-port protocol of Rexec.

BPA0153 RC_DAE_INTERNAL_ERROR

Explanation: Internal error. This is a follow-on error.

Severity: none

Module: DAE

System Action: A message containing this error is sent to the Rexec-client.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0154 RC_DAE_SETERRTRCCONFIG

Explanation: The settings from the Process Manager configuration file error logging and tracing are faulty.

Severity: Error

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0155 RC_DAE_UNDO_UPDATE_LCS_ENTRY

Explanation: Undo a previous update of a LCS entry of SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in a inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function

BPA0156 RC_DAE_INVALID_SSP_ASMT_EXEC_FAILED_MSG

Explanation: Internal Error. The internal Process Manager message is not valid.

Severity: Error

Module: DAE

System Action: This message is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0157 RC_DAE_UNDO_UPDATE_SSP_ENTRY

Explanation: Undo a previous update of a SSP entry of the SSPT ADT failed.

Severity: Error

Module: DAE

System Action: Process Manager is in an inconsistent state.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Start the garbage collection function.

BPA0158 RC_DAE_INVALID_CONFIGURATION_ENTRY_AS_INT

Explanation: A configuration parameter entry in the Process Manager's daemon start-up configuration file contains an invalid integer specification. See the supplied group and parameter specification for details.

Severity: Warning

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Refer to the supplied documentation.

BPA0160 RC_DAE_RES_FILE_NOT_FOUND

Explanation: The configuration file given at the start-up of the Process Manager's daemon has not been found. See the supplied pathname with the message.

Severity: Error

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the given name of the configuration file and the access privileges.

BPA0161 RC_DAE_STOPGARBAGECOLLECTORTHREAD

Explanation: The stop of the Process Manager garbage collector thread could not be explicitly performed.

Severity: Error

Module: DAE

System Action: none

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0162 RC_DAE_STARTGARBAGECOLLECTORTHREAD

Explanation: The garbage collector thread has not been started.

Severity: Severe

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0164 RC_DAE_CHECKANDOPENIPCOMMUNICATION

Explanation: Process Manager has been not able to open communications.

Severity: Severe

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Check that a correct IP-address and port has been specified in the Process Manager configuration file.

BPA0165 RC_DAE_ABEND

Explanation: Process Manager has abnormally ended due to a previous error.

Severity: Severe

Module: DAE

System Action: Process Manager shuts down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0166 RC_DAE_WRONG_EFFUID

Explanation: Process Manager started with the wrong effective userid, which is reported with this message. It is expected that the Process Manager daemon is started with effective userid 0. The processes of the Process Manager might not have the correct permission if the effective userid is not 0.

Severity: Warning

Module: DAE

System Action: none

System Programmer Response: Start Process Manager with effective userid 0.

BPA0167 RC_DAE_GETENV

Explanation: The environment variable _BPA_IPC_BASE_PATH as not found. The service for system internal communication to the Process Manager is disabled.

Severity: Warning

Module: DAE

System Action: Process Manager Daemon does not listen to requests send by bpaStartProgram' call.

System Programmer Response: If system internal communication is required, export the variable _BPA_IPC_BASE_PATH= as configured in [BASE]BasePath and restart the Process Manager. No other action required.

BPA0168 RC_DAE_RECV_SOCKET_WITH_DATA

Explanation: This is an internal error. The client socket cannot be received. No client connection is possible. This connection is required and enables the direct communication between client process and the requested server process.

Severity: Error

Module: DAE

System Action: Return an error to bpaStartProgram caller

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0169 RC_DAE_RETURN_SSD_PID

Explanation: The process identifier of the just dispatched process can not be returned. The requested process was successfully dispatched.

Severity: Warning

Module: DAE

System Action: none

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA0201 RC_SSPM_INVALID_ADT_SSPM_START_MORE_SSPS_MSG

Explanation: Internal error. SSPM received a ADT_SSPM_START_MORE_SSPS_MSG and detected an error when parsing the message.

Severity: Error

Module: SSPM

System Action: The message is ignored. No additional SSPs are started.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0202 RC_SSPM_INVALID_ADT_SSPM_TERMSSPS_MSG

Explanation: Internal error. SSPM received a ADT_SSPM_TERMSSPS_MSG and detected an error when parsing the message.

Severity: Error

Module: SSPM

System Action: The message is ignored. No SSPs are terminated.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0203 RC_SSPM_INVALID_SSP_SSPM_EXIT_MSG

Explanation: Internal error. SSPM received a SSP_SSPM_EXIT_MSG and detected an error when parsing the message.

Severity: Error

Module: SSPM

System Action: The message is ignored. The SSP is not restarted.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0204 RC_SSPM_INVALID_ADT_SSPM_TERMSSPM_MSG

Explanation: Internal error. SSPM received a ADT_SSPM_TERMSSPM_MSG and detected an error when parsing the message.

Severity: Error

Module: SSPM

System Action: The message is ignored. The SSPM is not terminated.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0205 RC_SSPM_MESSAGE_HEADER_LENGTH

Explanation: Internal error. The SSPM received a message on its datagram socket. The message header length is not the length expected.

Severity: Error

Module: SSPM

System Action: The message is ignored and the SSPM waits for the next message on the datagram socket.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0206 RC_SSPM_MESSAGE_UNSUPPORTED

Explanation: Internal error. SSPM received an unexpected message that is not supported.

Severity: Error

Module: SSPM

System Action: The message is ignored and the SSPM waits for the next message on the datagram socket.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0207 RC_SSPM_OPEN_SOCKET_ERROR

Explanation: During start-up, the SSPM received an error attempting to open the internal datagram socket.

Severity: Error

Module: SSPM

System Action: The SSPM terminates.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error. Shutdown and restart Process Manager.

BPA0208 RC_SSPM_RECVFROM_ERROR

Explanation: An error occurred trying to receive messages from the datagram socket.

Severity: Error

Module: SSPM

System Action: The SSPM is no longer able to communicate. The SSPM terminates.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error. Shutdown and restart Process Manager.

BPA0209 RC_SSPM_START_SSP_ERROR

Explanation: An error occurred trying to start an SSP.

Severity: Error

Module: SSPM

System Action: The SSPM ignores the error and continues starting additional SSPs, if necessary.

System Programmer Response: Check the access rights of the SSP executable. Check the system configuration for the number of processes that can be concurrently active. If the error persists, contact your IBM Service representative.

BPA0301 RC_SSP_INVALID_LHT_SSP_REXEC_MSG

Explanation: Internal error. The SSP received a LHT_SSP_REXEC_MSG and detected an error when parsing the message.

Severity: Error

Module: SSP

System Action: The SSP closes the communication socket. The SSP is terminated and restarted.

BPA0302 RC_SSP_MESSAGE_HEADER_LENGTH

Explanation: Internal error. The SSP received a message on its socket. The message header length is not the length expected.

Severity: Error

Module: SSP

System Action: The message is ignored and the SSP waits for the next message on it's socket.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0303 RC_SSP_MESSAGE_UNSUPPORTED

Explanation: Internal error. The SSP received an unexpected message that is not supported.

Severity: Error

Module: SSP

System Action: The message is ignored and the SSP waits for the next message on it's socket.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0304 RC_SSP_NOT_IN_SAME_AS

Explanation: Internal error. The SSP is running in a different Address Space that the parent SSPM.

Severity: Error

Module: SSP

System Action: The SSP is terminated and not restarted.

System Programmer Response: Start the garbage collection function or shutdown and restart Process Manager.

BPA0305 RC_SSP_OPEN_SOCKET_ERROR

Explanation: An error occurred when the SSP tried to open the communication socket.

Severity: Error

Module: SSP

System Action: The SSP is terminated and is not restarted.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error. Check the system configuration parameters, shutdown Process Manager, and restart Process Manager.

BPA0306 RC_SSP_RECVFROM_ERROR

Explanation: An error occurred trying to receive messages from the communication socket.

Severity: Error

Module: SSP

System Action: The SSP is terminated and restarted.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error. Check the system configuration parameters and shutdown and restart Process Manager.

BPA0307 RC_SSP_EMPTY_PASSWORD

Explanation: No password is received.

Severity: Error

Module: SSP

System Action: User login will be rejected.

System Programmer Response: A password must be specified at client application login.

BPA0401 RC_SSPT_CONDVAR_LIST_ELEM_NOT_FOUND

Explanation: The SSPT could not find the condition variable, which was created for the login handler thread, waiting for a free SSP to become available.

Severity: Error

Module: SSPT

System Action: The login handler thread waiting for the condition variable times-out.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0405 RC_SSPT_RESOURCE

Explanation: A wrong or missing parameter within one of the following entries of the configuration file:

```
[SSPT]
[USER] // optional group
```

Severity:
1. Wrong syntax: error
2. Missing parameter syntax: error
3. Wrong content: warning

Module: SSPT

System Action: Defined defaults are used. Refer to “Process Manager Configuration file parameters” on page 16 for a description of the configuration file parameter and for possible defaults. Whenever Process Manager discovers wrong contents specified within the parameter, it tries to correct them. See also “Plausibility checks of configuration parameters” on page 21 for a description of those correction algorithms.

System Programmer Response: Correct the specified error within the configuration file (try the default values). Whenever Process Manager discovers incorrect settings it logs the correction in the error log. Use the corrected values to update your configuration file. Restart Process Manager.

BPA0406 RC_SSPT_STATUS

Explanation: A status problem was detected within the SSPT ADT. This could be caused by a lost message.

Severity: Error

Module: SSPT

System Action: Process Manager will retry.

System Programmer Response: Start the garbage collection function.

BPA0407 RC_SSPT_TYPE

Explanation: Internal Error. An incorrectly configured entry was detected within the table.

Severity: Error

Module: SSPT

System Action: Process Manager will retry.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0408 RC_SSPT_LCS_ENTRY_NOT_FOUND

Explanation: The LCS, which was referred by the last message could not be found. This can be caused by high system loads, with resulting long message delivery times.

Severity: Error

Module: SSPT

System Action: Process Manager will retry.

System Programmer Response:

- Start the garbage collection function
- If the problem occurs frequently, increase the value for:

```
[SSPT]
SSPWaitTime = xy // has to be greater than Trimmtime
Trimmtime   = yz
```

BPA0410 RC_SSPT_SSPM_ENTRY_EXISTS

Explanation: The Process ID already exists within the SSPT ADT.

Severity: Error

Module: SSPT

System Action: The entry is not inserted into the table.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0411 RC_SSPT_SSPM_ENTRY_NOT_FOUND

Explanation: The SSPM specified by the process ID within one of the last messages could not be found in the SSPT ADT. This can be caused by high system loads, with resulting long message delivery times.

Severity: Error

Module: SSPT

System Action: Process Manager will retry.

System Programmer Response: Start the garbage collection function. If the problem occurs frequently increase the value for:

```
[SSPT]
Trimmtime=xy
```

BPA0412 RC_SSPT_SSP_ENTRY_NOT_FOUND

Explanation: The SSP specified within the last message could not be found within the SSPT ADT. This can be caused by high system loads, with resulting long message delivery times.

Severity: Warning

Module: SSPT

System Action: An insert of a new SSP will be done, whenever possible. An update or delete information specified within the last message sent to Process Manager will not be used to update the ADT.

System Programmer Response: Start the garbage collection function. Increase values for:

```
[SSPT]
SSPWaitTime=xy // Must be greater than TrimmTime.
TrimmTime  =yz
```

If the error persists, contact your IBM Service representative

BPA0413 RC_SSPT_SSPTABLE_INVALID

Explanation: Internal error.

Severity: Error

Module: SSPT

System Action: The function is not performed.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0415 RC_SSPT_INITIALIZED

Explanation: Internal error. The initialize function of the SSPT was called more than once.

Severity: Warning

Module: SSPT

System Action: The second function call is ignored.

System Programmer Response: No action required.

BPA0416 RC_SSPT_NOT_INITIALIZED

Explanation: Internal error. A SSPT ADT access function was called before the SSPT was initialized.

Severity: Error

Module: SSPT

System Action: The function is not performed.

System Programmer Response: Restart Process Manager. If the error persists, contact your IBM Service representative.

BPA0417 RC_SSPT_TRIMMING_ERROR

Explanation: An error occurred during positive or negative trimming, which is automatically done by the SSPT ADT. See the previous error messages for a more detailed description of the problem.

Severity: Error

Module: SSPT

System Action: Positive or negative trimming could not be performed at this time.

System Programmer Response: Observe the previous error messages. This problem can be caused by a high system load with resulting high resource consumptions. If the problem persists after restarting Process Manager, look at the system load. It may not be possible to run as many processes on your server as clients connecting to it.

BPA0418 RC_SSPT_PID_MISMATCH

Explanation: Internal Error. The PID specified within the last message does not match with the PID specified within a prior call.

Severity: Warning

Module: SSPT

System Action: The SSPT ADT will not use this different PID.

System Programmer Response: Start the garbage collection function.

BPA0419 RC_SSPT_ASID_MISMATCH

Explanation: Internal error. The address space ID (ASID) specified within the last message does not match with the ASID specified within a prior call.

Severity: Warning

Module: SSPT

System Action: The SSPT ADT will not use this different ASID.

System Programmer Response: Start the garbage collector with the command `s_gc`.

BPA0420 RC_SSPT_UNDO_IMPOSSIBLE

Explanation: Internal error. The Process Manager daemon called an undo function of the SSPT ADT. This function could not be executed, because there is no valid action to be undone.

Severity: Warning

Module: SSPT

System Action: The function is not executed. The system will retry.

System Programmer Response: No action required.

BPA0421 RC_SSPT_ALL_SSP_IN_USE

Explanation: Internal error. Process Manager tried to start a new SSP during positive trimming. An error occurred within the function calculating which SSPM can start one or more further SSPs. This SSPM table entry cannot hold more SSPs.

Severity: Error

Module: SSPT

System Action: The SSPT will not start any SSP under the control of this SSPM. The trimm function is not executed.

System Programmer Response: Start the garbage collector function.

BPA0422 RC_SSPT_INVALID_SERVERNAME

Explanation: No or incorrect ServerName.

Severity: Warning

Module: SSPT

System Action: None

System Programmer Response: Check configuration of the client application.

BPA0423 RC_SSPT_UNABLE_RESTART_SSPM

Explanation: An unexpected termination of a SSPM was detected. Normally a new SSPM would be started for this SSPM, but after the configured number of `MaxRestartSSPM` restart trials, no new SSPM could be started. This can normally only happen during heavy system load.

Severity: Warning

Module: SSPT

System Action: The Process Manager found that at least one other SSPM is running and will continue to run.

System Programmer Response:
- Check the value for `MaxRestartSSPM`.
- Check with `ps -e | grep bpasspm` how many SSPMs are running, or check how many clients are working with Process Manager. Perhaps too many clients tried to start programs on the server.
- Restart Process Manager.

BPA0424 RC_SSPT_ERROR

Explanation: An error occurred within the SSPT ADT. This is always a follow-on error. This message is logged.

Severity: Depends on the severity of the prior error message.

Module: SSPT

System Action: Depends on the first error causing this follow-on error.

System Programmer Response: See the description of the first error in the log, which will describe the error in more detail.

BPA0425 RC_SSPT_SSPM_PRESTART

Explanation: No SSPM or less than the configured SSPMs (specified by the Configuration file parameter `PrestartedAS`) could be started during Process Manager start-up. The number of successfully started SSPMs is specified within this error message.

Severity: Severe

Module: SSPT

System Action: If no SSPM could be started, Process Manager will shut down. If at least one SSPM was successfully started, Process Manager will remain started. The positive trimming function, which is automatically performed by Process Manager, will start more SSPMs, as needed.

System Programmer Response:
1. If **no** SSPM was started:
   a. Check the file permission bits of the SSPM executable (`bpasspm`). Check if the program is executable by the user rights of the userid which started Process Manager.
   b. Verify that the executables are specified within the `PATH` environment variable.
   c. Check the number of MAXPROCUSER in SYSTEM.PARMLIB(BPXPRMxy)
2. If less than the configured SSPMs were started:
   a. The operating system (system load) or hardware is not able to create more than the currently running SSPM's.
   b. Decrease the number specified for
   ```
   [SSPT]
   PrestartedAS = xy
   ```

BPA0426 RC_SSPT_INVALID_ENTRY

Explanation: Internal Error. Problem with consistency of the SSPT.

Severity: Error

Module: SSPT

System Action: The system will retry.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0427 RC_SSPT_UNABLE_TO_START_ANY_SSPM

Explanation: The SSPT is not able to start any SSPM. When the system is not able to restart at least one of SSPM, this error message is reported.

Severity: Severe

Module: SSPT

System Action: The Process Manager daemon ends.

System Programmer Response: Find the reason why all SSPM's ended. For a list of possible reasons see: message number BPA0425. Restart Process Manager.

BPA0428 RC_SSPT_REUSED_FAILED_SSP_ENTRY

Explanation: The SSPT has changed the state of some SSP entries to FAILED, because they did not become available within the period specified by `TrimmTime`. One of those entries is now reused for a SSP, which becomes available.

Severity: Warning

Module: SSPT

System Action: The new SSP is inserted into the table and will be used for future client requests.

System Programmer Response: If this message occurs frequently, increase the TrimmTime specified within the configuration file.

BPA0429 RC_SSPT_BAD_SSPM_TERM

Explanation: A SSPM unexpectedly terminated. All client processes within the same ASID are also terminated. This includes following processes:
- SSP
- RCS (server processes started for a client)
- LCS (server processes spawned by an RCS/LCS)

Severity: Warning

Module: SSPT

System Action:

- All LCS processes started in foreign address spaces from RCS processes controlled by this SSPM will be terminated.
- Process Manager tries to restart the SSPM several times.

System Programmer Response: If this problem occurs frequently, you might have problems with the maximum load on your machine. To avoid unnecessary Process Manager terminations, declare or increase the value of `MaxSSPMRestart` in group [SSPT] in the Process Manager configuration file.

BPA0431 RC_SSPT_EXIT_TABLE

Explanation: Internal error. The problem is encountered when trying to store the SSPM termination event.

Severity: Error

Module: SSPT

System Action: The information about this SSPM termination is lost.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA0432 RC_SSPT_TIMEOUT

Explanation: A time-out condition occurred. The daemon detected that a message was not received within the period of time, specified in `TrimmTime` or `SSPWaittime`. The problem can be caused by heavy system load (too many users connected at the same time.

Severity: Warning

Module: SSPT

System Action: The system will react differently, depending on the time-out situation:

Time-out of
: **Reaction**

SSPWaittime
: The system was not able to provide a free SSP to a client request within the specified time. The system will close the connection to the client.

Trimmtime
: An update message from a SSP was not received within the specified time period. The SSPT tries to start another one.

System Programmer Response: Determine the cause of this problem if the problem occurs frequently:
- Is there much network traffic on the machine caused additional by other programs?
- Is this situation typically caused by a peek of many users logging-in at the same time?

You can increase the following configuration parameters. This will allow more SSPs to be free to be assigned to an incoming client request. Increasing the time-out parameter will avoid unnecessary positive and negative trimming sequences.
- PrestartedAS
- PrestartedSSPPerAS
- MinFreeSSP
- StartAddSSP
- Increase the values for the timing parameter. SSPWaittime should be greater than Trimmtime.

BPA0433 RC_SSPT_LCS_ALREADY_EXISTS

Explanation: Internal error. A LCS entry with this PID was already inserted.

Severity: Error

Module: SSPT

System Action: This LCS entry will be ignored.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA0901 RC_SXT_NO_SOCKET_TRANSFERRED

Explanation: Internal Error. The expected socket has not been transferred.

Severity: Error

Module: SXT

System Action: Writes this error message to the Process Manager error log file.

System Programmer Response: Check the Process Manager error-log for related error messages and apply corrections, if possible. Otherwise, provide your IBM-representative with the error message.

BPA1001 RC_TRC_NOT_INITIALIZED

Explanation: This is a follow-on error and states that the trace facility could not be initialized.

Severity: Warning

Module: TRC

System Action: All trace statements within the executable will not be executed.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Restart the log daemon, if required, or restart Process Manager with active trace level=0.

BPA1002 RC_TRC_SIZE_EXCEEDS_MSG

Explanation: Internal error. The number of characters in the trace message is greater than the message queue message buffer.

Severity: Warning

Module: TRC

System Action: The trace record is truncated to fit in the message queue buffer.

System Programmer Response: No action required

BPA1004 RC_TRC_NO_KEYFILE_ENV

Explanation: Internal error. An environment variable, which is used to attach to the log daemon message queue, is missing.

Severity: Warning

Module: TRC

System Action: The trace facility will not be initialized

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA1005 RC_TRC_BUFFER_OVERFLOW

Explanation: A log string length exceeds the defined maximum length of 2048 bytes.

Severity: Warning

Module: TRC

System Action: The process where this error occurs is shut down.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA1101 RC_LOGD_NOT_INITIALIZED

Explanation: This is a follow-on configuration error during start-up of the Process Manager log daemon.

Severity: Error

Module: LOGD

System Action: The log daemon will shut down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Restart the log daemon.

BPA1102 RC_LOGD_ARGUMENT

Explanation: The log daemon was started with an incorrect argument.

Severity: Error

System Action: The log daemon will terminate the start-up process.

System Programmer Response: Start the log daemon with the following syntax: bpalogd <absolute configuration file>

BPA1103 RC_LOGD_RCH_RES_FILE_NOT_FOUND

Explanation: This is a follow-on error to InitResourceADT call. The displayed configuration file was not found.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Start the log daemon with the following syntax: bpalogd <absolute configuration file>

BPA1104 RC_LOGD_RCH_INITRESOURCEADT_FAILED

Explanation: This is a follow-on error to InitResourceADT call.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Restart the log daemon.

BPA1105 RC_LOGD_RCH_ERROR_LOG

Explanation: This is a follow-on error to GetResource call. The log daemon was started with a wrong or missing argument.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Start the log daemon with the following syntax: bpalogd <absolute configuration file>

BPA1107 RC_LOGD_RCH_FILE_RECORDS

Explanation: This is a follow-on error to GetResource call. The number of the possible file records of the trace log file is not in the allowed range.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Change the value ’FileRecords’ in the configuration file, group [TRACE]. Restart log daemon

BPA1108 RC_LOGD_RCH_BUF_RECORDS

Explanation: This is a follow-on error to GetResource call. The number for buffered trace records is not in the allowed range.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Change the value ’BufRecords’ in the configuration file, group [TRACE]. Restart the log daemon.

BPA1109 RC_LOGD_RCH_KEY_FILE_NAME

Explanation: This is a follow-on error to GetResource call. The returned key file name is unusable.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Change the name of 'KeyFileName' in the configuration file, group [TRACE]. Restart the log daemon.

BPA1110 RC_LOGD_RCH_LOG_FILE_NAME

Explanation: This is a follow-on error to GetResource call. The returned log file name is unusable.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Change the name of 'LogFileName' in the configuration file, group [TRACE]. Restart the log daemon.

BPA1111 RC_LOGD_RCH_TOSTDOUT

Explanation: This is a follow-on error to GetResource call. The flag 'ToStdout' in the configuration file is not accepted.

Severity: Error

Module: LOGD

System Action: The log daemon will post a RC_LOGD_NOT_INITIALIZED error.

System Programmer Response: Change the flag 'ToStdout' in the configuration file, group [TRACE]. Restart the log daemon.

BPA1112 RC_LOGD_WRONG_EFFUID

Explanation: This message occurs if the log daemon is started with effective user id unequal to zero.

Severity: Warning

Module: LOGD

System Action: None

System Programmer Response: Restart log daemon with root authority (uid=0).

BPA1113 RC_LOGD_SHUT_DOWN_ON_ERROR

Explanation: This error message occurs if the log daemon shuts-down due to an error. The previous error number is displayed.

If the previous error is a RC_RTS_MSGxxx, remove the log daemon message queue by restarting another log daemon.

Severity: Severe

Module: LOGD

System Action: The log daemon will shut down.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error. Restart the log daemon.

BPA1201 RC_FUX_BAD_ENVIRONMENT

Explanation: Internal Error. The process environment does not define an expected variable.

Severity: Error

Module: FUX

System Action: A notification to Process Manager is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative. Check the previous error return code(s) to obtain the reason of the error.

BPA1202 RC_FUX_CLOSESOCKET

Explanation: An z/OS UNIX domain socket cannot be closed as expected.

Severity: Error

Module: FUX

System Action: An z/OS UNIX domain socket survives in HFS until Process Manager is restarted. A notification to Process Manager is lost, which might result in an Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative. Check the previous error return code(s) to obtain the reason of the error.

BPA1203 RC_FUX_INITSUN

Explanation: An z/OS UNIX domain socket address cannot be initialized.

Severity: Error

Module: FUX

System Action: A notification to Process Manager is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative. Check the previous error return code(s) to obtain the reason of the error.

BPA1204 RC_FUX_OPENSOCKET

Explanation: An z/OS UNIX domain socket cannot be opened.

Severity: Error

Module: FUX

System Action: A notification to Process Manager is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative. Check the previous error return code(s) to obtain the reason of the error.

BPA1206 RC_FUX_NOTIFY_DAEMON_ABOUT_EXEC

Explanation: Notification of the Process Manager daemon about an EXEC failed.

Severity: Error

Module: FUX

System Action: A notification to Process Manager is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative. Check the previous error return code(s) to obtain the reason of the error.

BPA1207 RC_FUX_NOTIFY_DAEMON_ABOUT_FAILED_EXEC

Explanation: Notification of the Process Manager’s daemon about a failed EXEC failed.

Severity: Error

Module: FUX

System Action: A notification to Process Manager is lost, which might result in a Process Manager inconsistency.

System Programmer Response: Start the garbage collection function. If the error persists, contact your IBM Service representative.

BPA1208 RC_FUX_EXECV

Explanation: A call to C-RTS function execv () failed. Given with this message are the applied parameters ″path″ and ″argv[0]″.

Severity: Error

Module: FUX

System Action: The program specified did not execute as expected.

System Programmer Response: Check that the applied parameters are correct. Check the previous error return code(s) to obtain the reason of the error.

BPA1209 RC_FUX_EXECVE

Explanation: A call to C-RTS function execve () failed. The applied parameters ″path″ and ″argv[0]″ are displayed with this message.

Severity: Error

Module: FUX

System Action: The program specified did not execute as expected.

System Programmer Response: Check that the applied parameters are correct. Check the previous error return code(s) to obtain the reason of the error.

BPA1210 RC_FUX_EXECVP

Explanation: A call to C-RTS function execvp () failed. The applied parameters ″file″ and ″argv[0]″ are displayed with this message.

Severity: Error

Module: FUX

System Action: The program specified did not execute as expected.

System Programmer Response: Check that the applied parameters are correct. Check the previous error return code(s) to obtain the reason of the error.

BPA1211 RC_FUX_NOTIFY_DAEMON_ABOUT_SPAWN

Explanation: Notification of the Process Manager daemon about an ″spawnp″ failure.

Severity: Error

Module: FUX

System Action: The program specified with the spawn did not executed as expected.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA1212 RC_FUX_GETENV

Explanation: A value of a variable could not be read.

Severity: Error

Module: FUX

System Action: The system is not able to perform the function where the variable value is needed. Refer to the follow-on error.

System Programmer Response: Refer to the follow-on error.

BPA1301 RC_SHM_BAD_SHM_ATT

Explanation: An error occurred trying to attach the shared memory to the address space.

Severity: Error

Module: SHM

System Action: The SSPM ignores the error and continues with start-up. Follow-on application programs trying to attach to the shared memory will fail.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1302 RC_SHM_BAD_SHM_ID

Explanation: An error occurred trying to determine the ID of the shared memory.

Severity: Warning

Module: SHM

System Action: The SSPM ignores the error and continues with start-up.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1303 RC_SHM_CREATE_LOCK_ERROR

Explanation: An error occurred trying to create the mutex to protect the shared memory ADT.

Severity: Error

Module: SHM

System Action: The SSPM doesn’t start-up.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1304 RC_SHM_DESTROY_LOCK_ERROR

Explanation: An error occurred trying to delete the mutex used to protect the shared memory ADT.

Severity: Error

Module: SHM

System Action: Shutdown of the SSPM proceeds.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1305 RC_SHM_LOCKING_PROBLEM

Explanation: An error occurred trying to lock the shared memory table.

Severity: Error

Module: SHM

System Action: The shared memory table cannot be accessed by the application program.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1306 RC_SHM_TABLE_FULL

Explanation: Internal error. An error occurred trying to expand the shared memory table.

Severity: Error

Module: SHM

System Action: An entry cannot be inserted into the shared memory table.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1307 RC_SHM_TABLE_PROBLEM

Explanation: Internal error. An error occurred trying to insert an entry into the shared memory table.

Severity: Error

Module: SHM

System Action: An entry cannot be inserted into the shared memory table.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error.

BPA1308 RC_SHM_NO_ENV

Explanation: An error occurred trying to obtain the BSE variable from the process environment.

Severity: Warning

Module: SHM

System Action: The shared memory ID cannot be correctly resolved.

System Programmer Response: Check the previous run-time error return code to obtain the reason of the error. Set the application program environment correctly.

BPA1401 RC_GARC_RES_NOT_FOUND

Explanation: During start of the garbage collector, the configuration file could not be found.

Severity: Error

Module: GARC

System Action: The garbage collector will not be activated.

System Programmer Response: Restart the garbage collector with a corrected configuration file

---

BPA1402 RC_GARC_CREATEWAITSEM

Explanation: During start of the garbage collector, a semaphore could not be created.

Severity: Error

Module: GARC

System Action: The garbage collector will not be activated.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA1403 RC_GARC_CANT_ADD_CHILD_INDEX

Explanation: Internal error. The garbage collector could not completely build the parent-child relationship table. This is a follow-on error.

Severity: Error

Module: GARC

System Action: Refer to message number BPA1409.

System Programmer Response: Refer to message number BPA1409.

BPA1404 RC_GARC_CANT_FIND_CHILD

Explanation: Internal error. The garbage collector could not build the parent-child process relationship. This is an follow-on error.

Severity: Error

Module: GARC

System Action: Refer to message number BPA1409.

System Programmer Response: Refer to message number BPA1409.

BPA1405 RC_GARC_CANT_FIND_DPID

Explanation: Internal error. Garbage collector process inheritance table error.

Severity: Error

Module: GARC

System Action: Refer to message number BPA1409.

System Programmer Response: Refer to message number BPA1409.

BPA1406 RC_GARC_CANT_INIT_PIMT

Explanation: Internal error. The garbage collector process inheritance table could not be initialized.

Severity: Error

Module: GARC

System Action: Refer to message number BPA1409.

System Programmer Response: Refer to message number BPA1409.

BPA1407 RC_GARC_CHILD_NOT_FOUND

Explanation: Internal error. The garbage collector could not resolve the parent-child relationship.

Severity: Error

Module: GARC

System Action: Refer to message number BPA1409.

System Programmer Response: Refer to message number BPA1409.

BPA1408 RC_GARC_ERR_SSPT

Explanation: The garbage collector receives an error when using the SSPT functions. This is a follow-on error.

Severity: Error

Module: GARC

System Action: The garbage collector will not be activated.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA1409 RC_GARC_FILLPIMTABLE

Explanation: The garbage collector could not build the process inheritance model table. This is a follow-on error.

Severity: Error

Module: GARC

System Action: The garbage collector will not be activated.

System Programmer Response: If the error persists, contact your IBM Service representative.

BPA1410 RC_GARC_GETSSPTPIDS

Explanation: The garbage collector received an error from a server shell process table function. This is a follow-on error.

Severity: Error

Module: GARC

System Action: The garbage collector will not be activated.

System Programmer Response: Check the previous error return code(s) to obtain the reason of the error.

BPA1411 RC_GARC_LOST_CHILD

Explanation: Internal error. The garbage collector found an error in the SSPT regarding the parent-child relationship.

Severity: Error

Module: GARC

System Action: The garbage collector failed.

System Programmer Response: Restart the garbage collector later. There may be pending messages for the server shell process table.

BPA1412 RC_GARC_SSPT_V_PIM

Explanation: Internal error. The garbage collector found an error in the SSPT regarding the parent-child relationship.

Severity: Error

Module: GARC

System Action: The garbage collector fails.

System Programmer Response: Restart the garbage collector later. There may be pending messages for the server shell process table.

BPA1413 RC_GARC_IS_ALREADY_STARTED

Explanation: There is a running garbage collector thread. Only a single garbage collector thread is allowed. A follow-on attempt will fail.

Severity: Error

Module: GARC

System Action: The garbage collector start-up will be ignored.

System Programmer Response: No action is required.

BPA1416 RC_GARC_WAIT_SEMAPHORE

Explanation: The garbage collector thread could not de-activated, due to a previous error.

Severity: Error

Module: GARC

System Action: The garbage collector thread will terminate.

System Programmer Response: Restart Process Manager.

BPA2001 to BPA2088 RC_RTS_xxxx

Explanation: An error occurred in function call xxxx(), where xxxx is the function name. The errno and errno2() functions are evaluated and provide detailed information, depending on the individual function call return code. This information is appended to the individual error message.

Severity: Varies

Module: All

System Action: The function call xxxx will not be performed.

System Programmer Response: Resolve the run-time system error with the help of errno and errno2() information. Refer to *z/OS C/C++ Run-Time Library Reference* and *z/OS UNIX System Services Messages and Codes*. Restart Process Manager, if necessary.

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User’s Guide*, and *z/OS ISPF User’s Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

© Copyright IBM Corp. 1998, 2002

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 1998, 2002

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This publication primarily documents intended Programming Interfaces that allow the customer to write programs to obtain services of UNIX.

This publication also documents information that is NOT intended to be used as Programming Interfaces of z/OS UNIX. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

NOT Programming Interface information

End of NOT Programming Interface information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:
- DB2
- e
- IBM
- IBMLink
- Library Reader
- MVS
- MVS/ESA
- RACF
- Resource Link
- z/OS
- z/OS.e
- zSeries

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

© Copyright IBM Corp. 1998, 2002

Readers' Comments — We'd Like to Hear from You

z/OS
UNIX System Services
Connection Scaling Reference for iBaanERP Solutions

Publication No. SA22-7809-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SA22-7809-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape **Please do not staple** Fold and Tape

SA22-7809-02

Cut or Fold
Along Line

Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

SA22-7809-02