Component Broker for Windows NT and AIX

# Problem Determination Guide

*Release 2.0*

Component Broker for Windows NT and AIX

# Problem Determination Guide

*Release 2.0*

**Note!**

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

**First Edition (December 1998)**

This edition applies to Release 2.0 of IBM Component Broker and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

# Contents

# Notices

This publication was developed for products and services offered in the U.S.A. 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 in this publication. 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
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

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 DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS 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 publication 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 document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time without notice.

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
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

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 document and all licensed material available for it are provided by IBM under terms of the IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This document may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## Trademarks and Service Marks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

AIX
CICS
DB2
IBM
IMS
MVS/ESA
OS/2
PowerPC
VisualAge

AFS and DFS are trademarks of Transarc Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle and Oracle8 are registered trademarks of the Oracle Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

# About This Book

IBM Component Broker is a middleware product that can be used to design, develop, and deploy distributed object-oriented applications.

This book is intended for AIX and Windows NT Component Broker application developers and administrators who are responsible for diagnosing run-time problems. It provides information to help you resolve problems and, when required, report problems to IBM.

To use this book, you should be familiar with system and application programming, and the Component Broker's System Management tasks. You should also be familiar with the other systems used with the Component Broker at your site, for example, AIX, Windows NT, DCE, DB2, Oracle, CICS, and IMS.

**Related Information**
For AIX and Windows NT Component Broker installation problem determination information, refer to the *Planning, Performance, and Installation Guide.*

For OS/390 Component Broker run-time problem determination information, refer to the *OS/390 Component Broker Messages and Diagnosis Guide*.

**How This Book Is Organized**
The information in this book is organized into the following chapters:

Chapters 2 - 15 describe Component Broker logs, traces, and tools that you can use for run-time problem determination.

Chapters 16 - 18 provide information to help you diagnose Component Broker run-time problems and report problems to IBM



The Appendices provide sample logs, communications reference data, minor codes and error messages.

# Chapter 1. Problem Determination Information

IBM Component Broker provides many run-time information sources to help you perform problem determination. These information sources primarily consist of error logs and trace logs. Depending on where you are in the application development's life cycle, you may choose to use some of the information sources and not others. This book describes some of the error logs and trace data so that you can use them to diagnose run-time errors and speed up the problem determination process.

The following information sources and traces may help you diagnose errors and failures:



Some of these trace logs are automatically generated, while others require you to stop the server to set the trace and then restart the server to enable the capturing of trace data. Depending on where you are in the application's life cycle, you may be able to set or use some of these traces and not others. The following table depicts four stages of the Component Broker application's life cycle when you may encounter run-time problems: unit test, integration, deployment, and operation, and it summarizes the type of information sources you would use to help you diagnose these problems at each stage.

| **Information Sources: Traces/Trace Logs** | **Component Broker Application** | | | |
|---|---|---|---|---|
| | **Unit Test** | **Integration** | **Deployment** | **Operation** |
| **Activity Log** | Yes | Yes | Yes | Yes |
| **Error Log** | Yes | Yes | Yes | Yes |
| **Event Log** | Yes | Yes | Yes | Yes |
| **Agent Trace Log** | Yes | Yes | Yes | Yes |
| **OLT** | Yes | Yes | No | No |
| **Transaction Trace Log** | No | No | No | No |

| | | | | |
|---|---|---|---|---|
| **Query Service Trace** | Yes | Yes | No | No |
| **Cache Service Trace** | Yes | Yes | No | No |
| **ORB Request Trace** | Yes | Yes | No | No |
| **ORB Communication Trace** | Yes | Yes | No | No |
| **XYZ Trace** | Yes | Yes | No | No |
| **Security Service Trace** | Yes | Yes | No | No |
| **APPC Communications Trace** | No | Yes | No | No |

**RELATED REFERENCES**

Problem Determination (*Planning, Performance, and Installation Guide*)

# Chapter 2. Activity Log for Problem Determination

The activity log captures events that show a history of Component Broker Connector services' activities. Some of the entries in the log are informational, while others report on system exceptions, such as returned CORBA exceptions.

When you encounter Component Broker run-time errors, you will often find it useful to read the activity log and try to diagnose the problem yourself. When you require assistance from IBM to help you diagnose Component Broker problems, you will be asked to provide the formatted activity log output to IBM.

### Location of the Activity Log

There is one activity log for each host machine. The activity.log file resides in the Component Broker's service subdirectory. If the file does not exist, Component Broker automatically creates the file for you.

### Formatting the Activity Log for Reading

The log in the service directory must be formatted before you can read its contents. To format the file, you can use either of the following interfaces:

- The System Manager user interface to browse the log's entries. (Expand the host image under **Host Images** to find **Activity Log Images** and then **Latest**. Right click on **Latest** and select **Browse**.)
- The showlog utility to pipe the formatted output to a file which you edit or browse, for example,

  ```
  showlog activity.log -debug > showlog.out
  ```

  **Note**: Run the showlog utility on the host that had the error to get the optimum substitution values in the formatted output file.

In most situations, rather than using the System Manager user interface, you would probably use the showlog utility to format the activity.log file into a readable output file for browsing.

### Description of the Activity Log Entry

Each entry in the activity log has the following fields:

- **ComponentId**
  A numeric value assigned to each component in Component Broker. For example,

  ```
  ComponentId: 131175
  ```

- **ProcessId**
  The process number under which the server is running; this is the ID by which the operating system knows the process. For example,

  ```
  ProcessId: 491
  ```

- **ThreadId**
  The thread ID under which the object placed the event in the activity log. This is the thread ID by which the operating system knows the thread. For example,

  ```
  ThreadId: 601
  ```

- **FunctionName**
  The name of the function that placed the information in the activity log. This may not be very useful for you. For example,

  ```
  FunctionName:
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
  ```

- **ProbeId**
  This is typically the line number in the source file that has the function that placed the entry in the event log. However, some components may use this field as a probe ID and assign it some other value. For example,

  ```
  ProbeId: 4024
  ```

- **SourceId**
  The first number identifies the version of code that is running; the second entry is the location of the source code in the library system from which the code was built. This information may not be very useful to you. For example,

  ```
  SourceId: 1.3 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
  ```

- **Manufacturer**
  IBM
- **Product**
  Component Broker
- **Version**
  The Component Broker version
- **SOMProcessType**
  The type of process that placed this event in the activity log:
  - **1** for client process
  - **2** for ORB daemon
  - **5** for server process
    Individual types of servers (name and different application servers) are not distinguished.
- **ServerName**
  If this is a server process, the name of the server, for example,

  ```
  ServerName: gordian Name Server
  ```

- **clientHostName**
  The host name of the client if security is enabled for the server process.
- **clientUserId**
  The client user ID if security is enabled for the server process.
- **TimeStamp**
  Date and time when the event was placed in the activity log. The TimeStamp entry is unique, for example,

  TimeStamp: 12/5/97 14:08:25.784411745
- **UnitOfWork**
  The field contains the identification for the original request. The name is in the format, *nnnnn*:*hhhhh*, where:
  - *nnnnn* : is a random number
  - *hhhhh* : is the name of the host where the original request originated.

  For example,

  ```
  UnitOfWork: 12455:fred
  ```

  If a request is forwarded to another server as part of the original request, the unit of work from the original request is used. This enables the clients to track the work that is done as part of an original request. **Note**: You may find the UnitOfWork information useful when you are trying to find related entries in the activity log or when you are debugging problems across multiple machines.
- **Severity**
  The severity of the problem. The possible values are:
  - **1** Error

- **2** Warning
- **3** Informational

- **Category**
  The category of the failure. The possible values are:
  - **1** Error
  - **2** Activity
  - **3** Trace
  - **4** Trace Data
  - **5** Performance
- **FormatWarning**
  A numeric value indicating that the replacement text of the message was not in the correct form. This will be a non-zero value when an attempt to place the replacement text in the Primary or Extended Message was not correct.
- **PrimaryMessage**
  In the case that the entry was placed in the activity log as part of an exception, the PrimaryMessage field contains the objectName::methodName(parameter list):lineNumber and the type of the exception (this should always be CORBA::exception) and the specific exception that was raised.

  This field contains essential information for problem determination. The field indicates one of the following:
  - **Throw of exception**
    The entry indicates that the listed function determined that an exception should be thrown. Look at the **ExtendedMessage** information to help you identify the cause of the problem. This is a category 1 (Error) entry.

    ```
    PrimaryMessage: The function
    IBOIMLocalToServer_ICDSDataObjectImpl::getStringByName(const char*):1011
    raised CORBA exception
    IBOIMLocalToServer::ICDSDataObject::IAttributeNotFound, error code is
    0x0 0.
    ```
  - **Reraised Exception**
    The entry indicates that an exception was received and reraised. This entry allows you to trace related entries in the activity log. This is a category 1 (Error) entry. For example,

    ```
    PrimaryMessage: The function
    IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const
    ByteString&):4054 reraised CORBA exception CORBA::INTERNAL.
    ```
  - **Mapping of Exception**
    The entry shows the exception that was received and the new exception that was raised. This is useful when tracking a specific exception through the activity log. This is a category 1 (Error) entry. For example,

    ```
    PrimaryMessage: The function
    IBOIMExtLocalToServer_IDataObjectBase_Impl::retrieveFromDataStore():450
    received CORBA exception IBOIMException::IDataObjectFailed and raised
    CORBA exception CORBA::PERSIST_STORE.
    ```
  - **Activity Entry**
    An activity entry provides information. If the entry appears during an exception path, it contains information to help determine the cause of the problem for which the exception is being raised. If the entry is not on an exception path, the entry simply provides information as to the state of the server. This is a category 2 (Activity) entry. For example,

    ```
    PrimaryMessage: The function IPTransport::connect():936 reported an
    activity.
    ```

**Note**: If you see a minor code of this format, 0x4942*xxxx*, where *xxxx* are hexadecimal numbers, look in the “Appendix G. System Exceptions and Minor Codes” on page 217 for more information on the message.

- **ExtendedMessage**
  The extended message often provides additional information to pinpoint the exact cause of the failure.
- **RawDataLen**
  When raw data is provided as part of this log entry, the length of the raw data is shown in hexadecimal format. The raw data follows this entry and is shown in a 16-byte dump with the ASCII format of the data at the right of the dump. For example,

```
RawDataLen: 93
RawData:
0000 01 00 00 01 B5 01 00 00 - 00 01 4F 00 00 00 03 32 ..........O....2
0010 03 48 6F 73 74 49 04 67 - 6F 72 64 69 61 6E 03 53 .HostI.gordian.S
0020 6F 6D 53 65 72 76 49 04 - 67 6F 72 64 69 61 6E 20 omServI.gordian
0030 4E 61 6D 65 20 53 65 72 - 76 65 72 03 53 6F 6D 48 Name Server.SomH
0040 6F 6D 65 49 04 69 48 6F - 6D 65 4F 66 45 76 65 6E omeI.iHomeOfEven
0050 74 43 68 61 6E 6E 65 6C - 48 6F 6D 65 00          tChannelHome.
```

If the ORB communications trace is turned on, the trace data from GIOP packets is displayed in the RawData field.

**RELATED CONCEPTS**
Activity Log (*System Administration Guide*)

**RELATED REFERENCES**
“Reading the Activity Log”
“HandleSignal - General Page Fault (GPF) Exception” on page 11
“Hints and Tips: Activity Log” on page 103
“showlog Utility” on page 68
“Chapter 11. ORB Communication Trace” on page 41

**RELATED TASKS**
Display the Activity Log (*System Administration Guide*)

## Reading the Activity Log

Since the activity log is an accumulation of information, it always contains extraneous information. In most instances, lower level components decide to throw an exception and to place an entry in the activity log. The caller of the lower level component may be prepared to handle the exception and to absorb it and continue processing on a normal code path. The entry that the lower level component placed in the activity log remains in the log.

In some situations, components place warnings in the activity log to warn you of potentially dangerous situations. Sometimes these entries can help you determine the exact cause of the problem.

The first step in reading the activity log is to isolate the original exception that caused the failure. This is essential in identifying the cause of the problem.

The entries in the formatted activity log are listed from the oldest entry to newest. The oldest entries in the log probably do not pertain to the failure. Refer to the “Hints and Tips: Activity Log” on page 103 for suggested ways to reduce the

information in the activity log. When reading the formatted activity log, you need to identify the group of entries that are related to the problem or error that you are diagnosing. This *group* of entries form a *bracket*, as follows:

- **The start of the bracket**
  Initial failure, which is a single entry in the log
- **Results of the initial failure**
  A number of entries in the log
- **The end of the bracket**
  Last result of the failure, which is a single entry in the log

When reading the formatted activity log, start with its last entry and then work backwards, reading the previous entry and then the one before. The last entries of the log may not be related to the failure, for example, there may be entries made by the ORB, by the name server, by the client, or by other requests on application servers. Follow these steps to find the bracket of entries in the activity log that pertains to the error that you are diagnosing:

1. **Identify the end of the bracket**
   Reading backwards entry by entry, try to determine if the entry is related to the problem that is being investigated. A description of the first entry to look for in the activity log is dependent on the symptoms of the failure, and they are described in the activity log samples. (Hint: When reading backwards, you can look for the first non-error message. Back up to the last entry that you have read and that error entry may be the *end of bracket* entry for the problem that you are diagnosing.) Once the entry related to the failure has been identified, you have identified the *end of the bracket*.
2. **Find relevant entries**
   The next step is to examine each previous entry to see if it is related to the failure.
3. **Find the initial failure**
   Entries which are listed previous to the bottom of the bracket will need to be examined to determine which is the initial failure.

To describe the technique, review sample logs of these five basic failures that can occur:

- “Activity Log Sample 1: Server Does Not Start” on page 8
- “Activity Log Sample 2: Client Application Receives an Expected Error” on page 8
- “Activity Log Sample 3: Client Application Receives an Error And Server Dies” on page 9
- “Activity Log Sample 4: Client Application Receives an Error” on page 10
- “Activity Log Sample 5: Client Application Receives an Error” on page 10

If the ORB communication trace is turned on, the trace data is displayed in the Raw Data field.

Smaller activity logs may speed up your problem determination process. Refer to “Hints and Tips: Activity Log” on page 103 for more information on formatting the activity log before deleting it and rerunning your application.

**RELATED REFERENCES**

“Chapter 2. Activity Log for Problem Determination” on page 3
“HandleSignal - General Page Fault (GPF) Exception” on page 11
“Chapter 11. ORB Communication Trace” on page 41

## Activity Log Sample 1: Server Does Not Start

**Symptom:** The application server (this includes the name server) does not start.

**Finding the End of the Bracket**
In this case, the activity log will contain an entry from server startup indicating that the server failed to start. Locate this entry in the activity log:

```
ExtendedMessage: Server myServer failed
```

*myServer* is the name of the server which was attempting to start. Remember to start from the bottom of the activity log and work backwards.

**Activity Log Sample 1**
Refer to the “Activity Log Sample 1” on page 115. From the bottom of the activity log, work backwards and look for the first entry indicating that the server, fred, did not start. This is the end of the bracket.

Each entry in the activity log prior to the end of bracket entry must be examined to determine if it is the initial failure or if it is the result of the initial failure. Note that the related entries are for the same server. The initial error should be a Severity 1 error. The entries that form the start and end of the bracket are highlighted in bold in the sample.

**Recovery**
If you refer to “Troubleshooting Server Activation Problems” on page 71, you can find possible causes for this problem. In this example, the failure is a result of not starting the ORB daemon. Use the System Manager user interface to start the daemon.

**RELATED REFERENCES**
“Reading the Activity Log” on page 6

## Activity Log Sample 2: Client Application Receives an Expected Error

**Symptom**
The client application receives the exception, IManagedClient::INoObjectWKey. The client application may be coded to recover from this situation. Specifically in this case, the application can either display an error message stating that the requested information could not be located, or the application can create the object associated with the information.

**Finding the End of the Bracket**
The end of bracket entry should be the one that corresponds to the exception, IManagedClient::INoObjectWKey.

**Activity Log Sample 2**
Refer to the “Activity Log Sample 2” on page 141. This example is a common expected exception (IManagedClient::INoObjectWithKey) that you would handle in your client application. The recovery is to either create such an object, or to have the client application report back to the user that the object does not exist.

This sample activity log shows an expected exception when findByPrimaryKeyString is run against the home. Find the log entry dealing with the findByPrimaryKeyString method on the home.

The run time can log information. This is what the home does in the second to last entry in the log. Note that the home logs most of its internal state when a failure occurs. This can help you in the event that the findByPrimaryKeyString method fails when you expect it to succeed. Another important piece of data the home gives you in the log is the key that was passed in.

Some log entries do not list method names that are a part of the IDL interface for the framework. Because of the way logging works, the name of the method or function is reported in the log. If you see a method that does not have a name from the IDL interface that is described in the programming model, it is likely that a private implementation method was used by the run time.

Some activity log entries will simply reraise exceptions that they received from lower level calls. The fact that these reraised exceptions occur suggests that these entries are not the source of the problem. Sometimes, the run time remaps the exception it receives from a lower call to another exception which is defined on its interface. This is what happens when the run time receives an IBOIMException::IDataKeyNotFound exception from a user data object (first entry in the log). The run time converts this exception to INoObject withKey, which is the expected exception for the interface findByPrimaryKeyString (second entry in the log).

**Recovery**
The recovery for the problem is to determine if the nonexistence of the object is alright; in which case, the error can be ignored. If the nonexistence of the object is not alright, you must determine if the object should exist in the data base, and if the correct exception is being returned from the data object.

**RELATED REFERENCES**
“Reading the Activity Log” on page 6

## Activity Log Sample 3: Client Application Receives an Error And Server Dies

**Symptom**
The server starts. The client application is run, the client application hangs (if timeout is set to infinite) or ends with a CORBA::COMM_FAILURE. The application server is not running.

You can also receive unrecoverable exceptions in this situation. In this case, the server dies while running a client application. The client application deleted the same storage twice in the method, addBeneficiary, on the Policy object. There is no way the client application can recover from this error, because the server will abort.

**Activity Log Sample 3**
Refer to the “Activity Log Sample 3” on page 145. Whenever you see a handle signal in the log you have a serious problem (first entry in the log). See the description on “HandleSignal - General Page Fault (GPF) Exception” on page 11 for more information on this log entry. However, in this sample, because the addBeneficiary method was called remotely, the server caught the error and a dumpTargetInfo call is done. Note that the dumpTargetInfo entry indicates that the addBeneficiary method is the method that was being called on the Policy object. This provides a strong hint to look at the addBeneficiary code.

**Recovery**
The client application's code will have to be changed to fix the problem. Review addBeneficiary code and make necessary code changes.

**RELATED REFERENCES**
"Reading the Activity Log" on page 6

## Activity Log Sample 4: Client Application Receives an Error

**Symptom**
Sometimes an unexpected exception can be caught and handled without having to change your code. Often, this is the case when a configuration error occurs. The server starts. The client application is run, and the client application ends with a CORBA::UNKNOWN exception. The application server is still running.

**Activity Log Sample 4**
Refer to the "Activity Log Sample 4" on page 149. At the bottom of the log find an entry for a findByPrimaryKeyString call on a home returning a CORBA::UNKNOWN exception (last entry). Look at the previous entries to determine the source of the failure. Note that most of the log entries are simply reraising CORBA::UNKNOWN. This is an indication that these log entries are not the source of the failure, but merely the reporting chain from the failure to the outermost interface call, which, in this case, is findByPrimaryKeyString on the home.

You should then find a call to IRDBIMExtLocalToServer_IDataObject_Impl::retrieveFromDataStore (in the middle of the log marked in bold) which indicates that the data object is involved in the failure. If you continue to look in the previous entries, you will see an entry where it is apparent that a database connection to the Policy2 database cannot be made. You could then check to see if DB/2 is started.

**Recovery**
In this case, DB/2 was not started. When you start it and restart the application server, the application will run.

**RELATED REFERENCES**
"Reading the Activity Log" on page 6

## Activity Log Sample 5: Client Application Receives an Error

**Symptom**
The server starts. The client application is run, and the client application hangs (if timeout is set to infinite) or ends with a CORBA::NOTRANSACTION exception. The application server is still running.

**Activity Log Sample 5**
Refer to the "Activity Log Sample 5" on page 156. Another example of a recoverable unexpected exception occurs in this log. Note that the findByPrimaryKeyString call on the home failed (second last entry). Since CORBA::NOTRANSACTION is the exception, why is the transaction not available? Scanning upwards in the log also confirms that the transaction is not started (first entry in the log).

**Recovery**
Check your client application to verify that no transaction was started. If this is the case, the client application has to be changed and rebuilt.

**RELATED REFERENCES**
"Reading the Activity Log" on page 6

## HandleSignal - General Page Fault (GPF) Exception

In some instances, the activity log captures data as a result of receiving a hardware error. Component Broker has a signal handler registered with the operating system. The signal handler allows the failure to be trapped, and information about the process is captured. The failure is then turned into a C++ exception and raised in the handleSignal method.

A signal or hardware error can occur for a number of reasons, such as:

- Division by zero
- Dereferencing a null pointer
- Accessing memory which has already been freed

**Activity Log Entry That Has A General Page Fault Exception**
When this error occurs, the signal handler first writes an entry in the activity log. Refer to the "Activity Log Showing General Page Fault Exception" on page 161. The entry shows:

- **Usual Activity Log Entry Information**

```
ComponentId:131175
ProcessId:475
.
.
.
FormatWarning: 0
PrimaryMessage:The function handleSignal(int,int,CONTEXT*):608 reported
  an error.
ExtendedMessage: A system error was detected - 40 SEG-VIOLATION (OS
  signal nbr: 2).
```

- **Type of Error Which Was Received**

```
Intel context flags:
i386/486 CONTROL INTEGER SEGMENT FULL FLOAT-POINT DEBUG-REG
```

- **Register Information for the Process**

```
Intel segment registers:
Gs:00000000 Fs:0000003b Es:00000023 Ds:00000023 Ss:00000023 Cs:0000001b
Intel GP registers:
Edi:068250f0 Esi:0225240f Ebx:068250f0 Edx:000009a4 Ecx:00000000
Eax:00000000 Ebp:02dbf42c Eip:064d61fa Esp:02dbf3fc
```

- **Current Instruction Pointer**

```
Error occurred at (Eip) :064d61fa, in source file: PolicyS.dll
```

- **Call Stack**

```
02dbf42c (062C9120 - PolicyDB2.dll)
02dbf48c (06399054 - PolicyC.dll)
02dbf4c4 (0639998A - PolicyC.dll)
02dbf584 (06485190 - PolicyS.dll)
02dbf5d0 (6B6859E7 - somorori.dll)
02dbf828 (6B6857F8 - somorori.dll)
02dbfc0c (6EACDCE5 - somsrsai.dll)
02dbfd80 (6EACFB5D - somsrsai.dll)
02dbff54 (504419F7 - cppobi36.dll)
02dbffa4 (5043F663 - cppobi36.dll)
02dbffb8 (77F04F2C - KERNEL32.dll)
```

- **Module List, and DLL Load Points**

```
Module list: size (load address) date-linked-GMT => DLL/EXE name
512000 (00240000) 1998/04/18 04:27:22 GMT => c:\IBMCPPW\BIN\CPPWM35I.dll
19456 (00360000) 1998/09/09 10:46:19 GMT => e:\e9836.01.lite\lib.nt\teceifi.dll
289280 (00400000) 1998/09/09 11:22:59 GMT => e:\e9836.01.lite\bin.nt\somsrsm.exe
4605952 (004C0000) 1998/09/22 19:17:03 GMT => e:\e9836.01.lite\lib.nt\somibe1i.dll
.
.
.
355088 (77F60000) 1997/04/11 20:38:50 GMT => C:\WINNT\System32\ntdll.dll
149264 (77FD0000) 1996/07/17 18:15:07 GMT => C:\WINNT\System32\WINMM.dll
271632 (78000000) 1997/01/23 07:07:13 GMT => C:\WINNT\system32\MSVCRT.dll
70656 (780A0000) 1997/01/23 05:27:47 GMT => C:\WINNT\System32\MSVCIRT.dll
```

- **Dump of the Storage**

```
Storage dump:Storage dump:
02DBF40C AA AA AA AA 40 AD 58 06 - AA AA AA AA AA AA AA AA ....@.X.........
02DBF41C 00 00 00 00 DC FF FF FF - 84 F4 DB 02 90 D0 53 06 ..............S.
.
.
.
02DBF50C A0 D8 23 06 78 D1 23 06 - 20 F8 DB 02 00 B0 CC 04 ..#.x.#........
02DBF51C 6B 16 41 50 40 00 00 00 - 34 72 52 50 00 00 00 00 k.AP@...4rRP....
```

After the entry has been placed in the activity log, the handleSignal method raises an exception, SOMRASOperatingSystemException (a subclass of an IException). The exception is caught in the nearest catch block, and error processing occurs on the code path.

**Determining Where the Failure Occurred in a handleSignal Dump**
When a handleSignal dump appears in the activity log, it is very important to determine where the failure occurred. In most instances the statement causing the failure is :

- The caller of the method (PolicyDB2.dll in the sample dump)
- The method where the signal occurred (PolicyS.dll in the sample dump)

There are two ways to determine the method that caused the failure:

- **Use the Debugger**
  If the caller or the DLL listed at the top of the stack is compiled with debug, it is possible to attach the debugger to the server and run the application that caused the failure. When the signal occurs, the debug will break (assuming that the options are set correctly), and the call stack can be examined to determine the statement that caused the error.
- **Translate the Stack**
  If the DLLs at the top of the stack are your DLLs, rather than using the debugger, you can translate the stack to identify the method that caused the failure. To do this, you will need the map files that were generated when you built the DLL. If the DLLs listed near the top of the stack are owned by Component Broker, you will not be able to use the debugger to help you diagnose the problem, and you will also require an IBM Representative to access the information that is required to process the stack. Unfortunately, by translating the stack, you cannot identify the exact statement that caused the failure in the DLL; you can only identify the method where the failure occurred.

**Calculating the Call Stack**
The map files for the DLLs that appear in the call stack are needed to calculate the method where the failure occurred. The map files for the som*xxxx*.dll modules are only available to IBM Support staff.

Each entry in the stack must be calculated. This sample only shows how to calculate the top entry in the stack. The process must be repeated for each entry. Refer to the “Activity Log Showing General Page Fault Exception” on page 161 when following these steps to calculate the call stack:

1. Identify the address you want to convert. Look at the **Error occurred** statement to find out the address where the error occurred.

   ```
   Error occurred at (Eip):064d61fa, in source file:PolicyS.dll
   ```

   In this example, the address is 064d61fa.
2. Identify the DLL. The DLL name is given by the source file statement. In this example, it is PolicyS.dll. Use this name when locating the map file.
3. Locate the DLL in the module list. Note the load point. The following is the load point information for PolicyS.dll:

   ```
   2835456 (06430000) 1998/09/28 15:20:32 GMT =>
                      e:\e9836.01.lite\lib.nt\PolicyS.dll
   ```

   The load point is 06430000. It will be used in a later calculation.
4. Perform the first calculation. To determine the offset in the PolicyS.dll where the failure occurred, the offset from the load point of the DLL will need to be calculated. The formula is:

   offset = instruction pointer - load point
   offset = 064d61fa - 06430000
   offset = A61FA

   The offset will be used to determine the location in the map file.
5. Locate the map file that corresponds to the DLL. These are located in the source tree on development drivers. Component Broker .map files are not available to you. For user DLLs, the **link** option must be specified at build time. See an example of the map file in the “HandleSignal - Compilation Map File Example” on page 164.
6. Find the offset. Find the line, **Image based at**, in the .map file. The offset addresses in the map file are located there. Note the value which the image is loaded. In most DLLs, the offset will be 400000.
7. Perform the second calculation. The offset from the previous calculation should be added to the value specified by the map file’s **Image based at** address.

   address = offset + image based at
   address = A61FA + 400000
   address = 4A61FA
8. Locate the address in the map file. Locate the method in the map file that contains the calculated address. Note that the map file only contains the starting address of the methods, so the exact address that you have calculated does not appear in the map file. In the example, the corresponding method in the map file is :

   004A6170 ?addBeneficiary__13PolicyBO_ImplFv
9. The name is in a mangled format. **CPPFILT** or a similar tool can be used to demangle the name.

**Interpreting the Handle Signal Activity Log for Your Code**

Use the following tips to help you determine if your code caused the exception:

1. Your DLL is given as the address where the error occurred, then your code caused the exception, for example:

   ```
   Error occurred at (Eip):064d61fa, in source file:PolicyS.dll
   ```

2. If your DLL is in the call stack and there are no Component Broker DLLs between the error point and your DLL in the call stack, then your code caused the exception. Component Broker DLLs all have the prefix "som". For example, the following information shows that the user code caused the error:

   ```
   Error occurred at (Eip):014c61fa, in source file:msvcrt.dll
   ```

   Notice that a system DLL is identified as the module that caused the error. The Call Stack shows your DLL at the top of the stack, as follows:

   ```
   01b9e824 PolicyS.dll
   03004a0300 somibs1i.dll
   03804a0300 somibe1i.dll
   ```

**RELATED REFERENCES**

“Chapter 2. Activity Log for Problem Determination” on page 3

# Chapter 3. Error Log for Problem Determination

The error log is a binary file that contains single entries for a failure. The log is automatically created for you. It is a subset of entries that are in the activity log. You can use the activity log instead of the error log for problem determination purposes.

### Location of the Error Log

The implementation of the error log is platform-specific:

- WIN On Windows NT, the error log is implemented using the NT event log.
- On other platforms, the error log is located in the same place as the activity log.

### Browse the Error Log

WIN For Windows NT, use the Administration facility to view its event log.

For other platforms, you can format the error log for browsing using the **showlog** command, as follows:

```
showlog error.log -debug > error.out
```

Use an editor to browse the contents of the error.out file.

### Interpreting the Error Log

The error log entries are identical to activity log entries. Refer to the “Reading the Activity Log” on page 6 for more information on how to interpret the entries.

**RELATED REFERENCES**

“Chapter 2. Activity Log for Problem Determination” on page 3

# Chapter 4. Event Log for Problem Determination

WIN The event log only exists for Windows NT Component Broker installations. You seldom have to look at the Windows Event Log for problem determination purposes, although some system exceptions (for example, lifecycle or RAS minor error codes) would ask you to look at the event log (for example, to see if you have DCE errors besides Component Broker errors). To view the Windows NT event log you can use the **Event Viewer**, which is one of the **Administration Tools (Common)** for Windows NT. You can browse the event log without interrupting other processes on the host. All messages stored by the Component Broker product can be distinguished by the string, **CBConnector**, in the **Source** column in the Windows NT event log.

**RELATED TASKS**

Display Component Broker Events Using the Tivoli Event Manager (*System Administration Guide*)

Display Component Broker Messages in the Windows NT Event Log (*System Administration Guide*)

**RELATED REFERENCES**

"Appendix G. System Exceptions and Minor Codes" on page 217

# Chapter 5. Agent Trace Log for Problem Determination

The Component Broker System Management code uses its own tracing functions. Much of the System Management code has trace points at each function entry and exit points, and many System Management modules trace other execution details. By default, all exceptions are traced. When reporting problems to IBM, you may be instructed to customize this trace activity.

This topic provides the following information:



### Locating the Agent Trace Log

The agent trace log consists of several files with the following names:

- *executableName_systemName*.dct
  This is the trace "dictionary".
- *executableName_systemName*.tr1
  This is trace file 1.
- *executableName_systemName*.tr2
  This is trace file 2.

For example, for **bgmain** running on machine bumppo.austin.ibm.com, you will get:

- bgmain_bumppo.austin.ibm.com.dct
- bgmain_bumppo.austin.ibm.com.tr1
- bgmain_bumppo.austin.ibm.com.tr2

On a Component Broker install image, these files are in the data subdirectory, /var/CBConnector/data on AIX AIX, and d:\CBroker\data on WIN Windows NT. Trace records are collected in the *xxx*.tr1 file until it reaches the default maximum size of 100KB; then records are collected in the *xxx*.tr2 until it also reaches 100KB. These two files are swapped back and forth as they fill up. Control information is collected in the *xxx*.dct file.

**Note**: The agent trace logs are erased each time you stop and then restart the Component Broker Connector service. If the agent trace logs are required, you should *not* stop and restart the Component Broker Connector service until you have formatted these logs.

### Formatting the Agent Trace Log

To format the agent trace log, you use the **bgtrfmt** utility. If you have trace files, say, bgmain_mySystem.dct, bgmain_mySystem.tr1, and bgmain_mySystem.tr2, you would invoke:

```
bgtrfmt bgmain_mySystem > bgtrace.output
```

WIN When running on Windows NT, you will not be able to run **bgtrfmt** if the trace files are still open by the System Management executable that created them. You will have to either stop the executable or create a temporary directory and copy the three trace files to that directory. You may find the latter approach to be simpler.

### Browsing Agent Trace Log from the System Manager User Interface

Some messages in somsmmsg.msg instruct you to look at the agent trace log for details. You can also browse the agent trace log as follows:

1. Find the trace log under **Host Images...Agent Images...Trace Log Images...Latest**.
2. Right click on **Latest** and select **Browse**.

### Interpreting the Agent Trace Log

The agent trace log is most useful when you need to understand the sequence of events leading up to a failure, for example, if you are diagnosing System Management problems. Once you understand a little bit about a problem from the activity log, you can look at the agent trace log and see what events led to the failure. The trace output is tied to the source code and some of the output will not be very informative without the source. Here are two examples of agent trace logs. The notation, //..., means that text has been removed from the sample.

### Agent Trace Log Example 1

This example is the trace produced  for a re-activation of a small sample configuration. Here is an explanation of the fields:

```
433 655 09/11/98 15:32:04.510000 0 1806 bgmosvrimg.cpp ! 1 Server kaskan//...
|   |   |        |               | |    |              | | |
|   |   |        |               | |    |              | | Trace record
|   |   |        |               | |    |              | | text
|   |   |        |               | |    |              | Nesting level
|   |   |        |               | |    |              | (relevant for
|   |   |        |               | |    |              | ENTRY/EXIT traces)
|   |   |        |               | |    |             Thread switch indicator;
|   |   |        |               | |    |             if present, it means
|   |   |        |               | |    |             that this record is
|   |   |        |               | |    |             from a different thread
|   |   |        |               | |    |             than the previous record.
|   |   |        |               | |    File Name
|   |   |        |               | Line number in file
|   |   |        |               Trace level
|   |   |        Time stamp
|   |   Date stamp
|   Thread ID
Process ID
```

Here is the trace log for example 1:

```
433 562 09/11/98 15:30:37.796000 0 230 bgloadc.cpp ! 1 It took 0.40 sec//...
433 562 09/11/98 15:30:38.447000 0 366 bgmobaseroot.cpp 1 OpenEvent fai//...
433 562 09/11/98 15:30:38.447000 0 367 bgmobaseroot.cpp 1 This is most //...
433 614 09/11/98 15:30:48.511000 0 230 bgloadc.cpp ! 1 It took 0.10 sec//...
433 656 09/11/98 15:31:46.424000 0 209 bgmoutil.cpp ! 1 09/11/98 15:31://...
433 656 09/11/98 15:31:46.675000 0 209 bgmoutil.cpp 1 09/11/98 15:31:46//...
433 656 09/11/98 15:31:47.546000 0 209 bgmoutil.cpp 1 09/11/98 15:31:47//...
433 656 09/11/98 15:31:47.917000 0 209 bgmoutil.cpp 1 09/11/98 15:31:47//...
433 656 09/11/98 15:31:49.569000 0 65 phase2ActivateThread.cpp 1 _clien//...
433 656 09/11/98 15:31:49.569000 0 1701 configmo.cpp 1 09/11/98 15:31:4//...
433 655 09/11/98 15:32:04.510000 0 1806 bgmosvrimg.cpp ! 1 Server kaska//...
433 655 09/11/98 15:32:25.030000 0 1806 bgmosvrimg.cpp 1 Server testsrv//...
```

The most interesting records in this output sample are the two at the end, reporting the starting of the two servers.

### Agent Trace Log Example 2

Example 2 is a similar activation, but this time somsmsoi.dll has been removed. You can see that an exception is reported from bgloadc.cpp.

```
422 522 09/11/98 16:19:28.772000 0 230 bgloadc.cpp ! 1 It took 0.40 sec//...
422 522 09/11/98 16:19:28.792000 0 366 bgmobaseroot.cpp 1 OpenEvent fai//...
422 522 09/11/98 16:19:28.792000 0 367 bgmobaseroot.cpp 1 This is most //...
422 482 09/11/98 16:19:38.867000 0 230 bgloadc.cpp ! 1 It took 0.10 sec//...
422 482 09/11/98 16:19:38.867000 0 315 bgloadc.cpp 1 Null create fn
422 482 09/11/98 16:19:38.867000 0 316 bgloadc.cpp 1 DLL somsmso Class //...
422 482 09/11/98 16:19:38.887000 0 534 bgobjl.cpp 1 Stage : 509
422 482 09/11/98 16:19:38.887000 0 536 bgobjl.cpp 1 Object : <□kaskan□H//...
422 482 09/11/98 16:19:38.887000 0 537 bgobjl.cpp 1 Root : <HostI□kaska//...
422 482 09/11/98 16:19:38.907000 0 538 bgobjl.cpp 1 Exception data ...
  text: Failed to load dll : somsmso
  text: bh_exception
  in file: bgloadc.cpp
  at line: 317
  in function: ABHGObjectBuilder::createObject(const bh_string&,const b//...
  in file: bgobjl.cpp
  at line: 538
  in function: bh_cur_local::init_obj(bh_semdir*,bh_semdir*,const char*,//...
```

**RELATED REFERENCES**

“Chapter 1. Problem Determination Information” on page 1
“bgtrfmt Utility” on page 69

# Chapter 6. Object Level Trace for Problem Determination

Object Level Trace (OLT) allows you to debug both client and server code as if they were resident on a single machine. With OLT, you can debug from the client machine, the server, your development workstation, or any other machine on which you have installed the Component Broker Toolkit. You may find OLT useful when diagnosing some run-time problems.

However, you have to recompile your application for OLT debugging. If you want a quick simple method trace, you can run the ORB request trace. Also OLT's debugger ignores Managed Object Framework (MOFW) functions and steps over "glue code". To debug the "glue code", you should consider doing the XYZ trace that is described in “Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace” on page 51.

**RELATED CONCEPTS**

Object Level Trace Overview (*Application Development Tools*)

**RELATED REFERENCES**

“Chapter 10. ORB Request Trace” on page 39
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72

# Chapter 7. Transaction Service Log and Problem Determination

The Transaction Service creates a log for every server. It records information about in-flight transactions. When restarting an application after a failure, these logs are used to roll back transactions. **Warning**: Do not erase these logs because they are required for recovery. You would not read these logs for problem determination purposes.

### Transaction Service Log Files

The name of a server's transaction service log is assigned by the Transaction Service the first time the server is started, and the name appears in one of the messages written to the Component Broker's activity log when the server starts up. The name has the format, somtr*nnnn*, for example, somtr0000. The log consists of several files that have the file extensions .ctl, .csh, and .*nnn* where *n* is a number.

**RELATED CONCEPTS**

Transaction Service Log (*System Administration Guide*)

"Chapter 14. APPC Communications Trace" on page 61

# Chapter 8. Query Service Trace

When you get OO-SQL errors in the activity log, use the query service trace to help you diagnose these problems. This topic describes how to turn on the query service trace and interpret the trace output:



### Setting the Query Service Trace Level

To set the query service trace level, complete the following steps:

1. Display the System Manager user interface, and set the user-level to **Expert**.
2. Expand the **Host Images** folder.
3. From the pop-up menu of the Server Image for the application server that you are interested in, click **Edit**. This displays the Object Editor for the Server Image. **Note**: You do not have to stop and start the server to set this trace.
4. In the Object Editor window, click the **Object Services Trace** tab.
5. Set the **query service trace level** attribute value to **advanced**, to enable trace information to be recorded for SQL queries.
6. Optionally, if you want to see a trace of actual data flows from the datastore, you must also set on the cache service trace, by completing the following steps:
   a. Click the **Component Trace** tab.
   b. Set the **caching component trace level** attribute value to **advanced** (same as the **query service trace level** attribute).

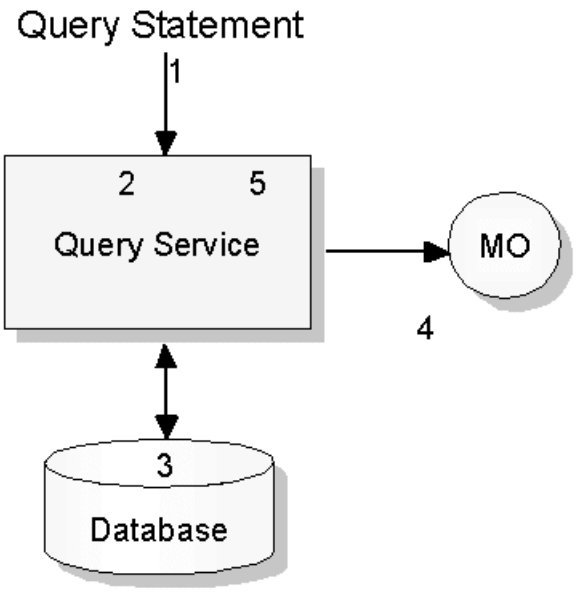
This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on formatting trace logs.

### Interpreting the Query Service Trace

To be able to work through query service trace information, you need to understand the processing flow within query service.

### The Flow of Control for a Pushdown Query

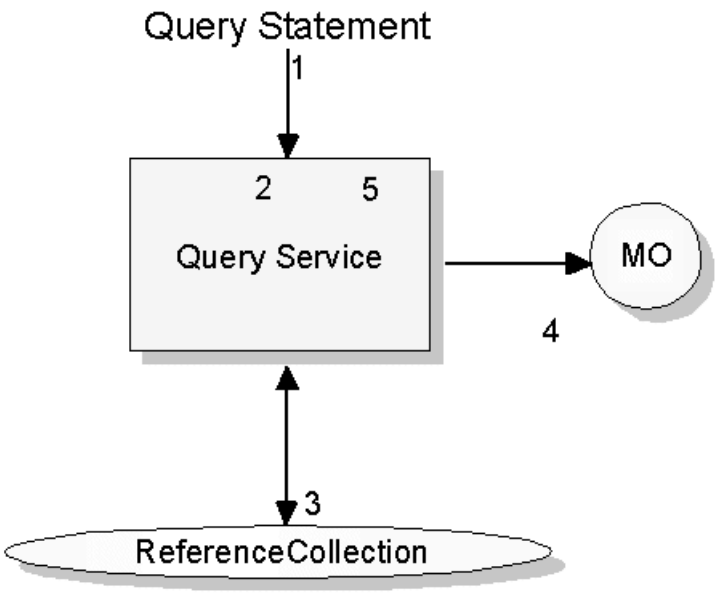
The following figure shows the flow of control for a pushdown query. This is a query over a queryable home or view collection that is translated into a database query. The flow of control is described after the figure.




1. A query statement is entered either via the queryable collection or query evaluator interface.
2. A pointer to the home collection is obtained either via the parameter list that is passed to the query evaluator or via the Name Service. The query is rewritten from OO-SQL to SQL. The Application Family configuration information is accessed via mapping information that describes how BusinessObject attributes map to tables and columns in the database.
3. The rewritten SQL query is executed against the database, and the database returns the result set. For certain types of queries, the iterator returned to the caller is an object wrapper for an SQL cursor. In this case data is fetched from the database as required by the iterator. For other iterators, the entire result set is fetched into memory during this step.
4. If the query involves returning object references, executing object methods, or accessing structs, then the result set must be transformed into a set of ManagedObjects. This is accomplished by calling the internalizeData() method on the DataObject interface of the DataObject contained within a ManagedObject.
5. Any residual predicates, sorting or aggregation is performed and an iterator to the final result set is returned to the caller.

**The Flow of Control for Query over a Reference Collection of a Managed Object**

The following figure shows the flow for a query over a reference collection of a managed object. The flow of control is described after the figure.




1. A query statement is entered via the query evaluator.
2. A pointer to the ReferenceCollection is obtained either via the parameter list that has passed to the query evaluator or via the Name Service.
3. An iterator and the interface name of objects contained in the collection is obtained from the reference collection. The InterfaceRepository is used to access a description of the interface Name and the query statement is validated.
4. For each MO in the collection, callMethodByName is used to dynamically call methods and access attributes on the MO.
5. A result set is constructed as a subset of the reference collection, and an iterator over the result set is returned to the caller.

### Examples of Query Service Trace Records

A formatted trace file generated by the showlog utility contains entries similar to the following examples.

### Example 1: Query Service Trace

The following trace record shows the query statement (Query =) and the query plan (Plan =) which contains the **select** statement executed against the database shown in bold. The keyword **_lazy_for** indicates that this is a demand driven iterator, and rows are fetched from the database as they are required by the iterator.

```
PrimaryMessage: The function
  osql::osql_exec_lazy(char*):1072 reported trace message.
ExtendedMessage: Query = "select ref a from orgMOHome a where
  a."dept"..id=2;"
  Plan = "_begin _plan
  _qes_tuple* _q1;
  _begin
  _i1 = _qes_tuple* _SQL DB2CS fksample [: select q1."ID", q1."A",
  q1."B", q1."C", q1."DEPT", q1."DTYPE" from org q1,
  org q2 where ( q2."DTYPE" = 'DEPARTMENT') and ( q2."ID" = 2) and (
  q2."ID" = q1."DEPT") and ( q1."DEPT" = 2) :]'org',
  'department';
  _lazy_for _all _q1 _in _i1 _do _begin
  _print _s2 ("1" _make_bo ("orgMOHome", _make_do
  ("orgDOImplDO_DAO", _make_dao ("orgPODAO_Table(orgPO_Alias)",
  "org", ((_qes_tuple*) _q1)->_c1(%_integer%), ((_qes_tuple*)
```

```
  _q1)->_c2(%_integer%), ((_qes_tuple*) _q1)->_c3(%_integer%), ((_qes_tuple*)
  _q1)->_c4(%_integer%), ((_qes_tuple*) _q1)->_c5(%_integer%), ((_qes_tuple*)
  _q1)->_c6(%_character%)))));
  _end
  _s0 = _all _s2 ;
  _end
  _end _plan
```

### Query Service Trace: Example Two

This trace record shows the call to buildFromData on the home collection. The home collection creates the MO and calls internalizeData on the DataObject passing to the DataObject the data values shown in the trace record. The value '-' is the SQLNULL value. Processing exceptions that occur after this point, may indicate a problem in the internalizeData() method of the DataObject.

```
PrimaryMessage: The function
  qes_eval_expr(qes_ptex*,osql_tsd*):3031 reported trace message.
ExtendedMessage: buildFromData arguments = "orgDOImplDO_DAO (char*: orgDOImplDO_DAO,
  dao (long: 6, long: 2, long: 6, - , long: 2, char*: PERSON) ) "
```

### Query Service Trace: Example Three

This entry in the following trace example shows the rewritten SQL as:

```
 _SQL DB2NT fksample [:  select  q1."ID",  q1."X",
       q2."ID",  q2."Y" from person1 q1, person2 q2
            where  ( q1."ID" =  q2."ID")  :]
```

**DB2NT fksample** means that the target database is a DB2 for Windows NT database whose dbname is **fksample**.

The actual SQL statement is:

```
 select  q1."ID",  q1."X",  q2."ID",  q2."Y" from
      person1 q1, person2 q2 where  ( q1."ID" =  q2."ID")
```

The original OO-SQL statement is:

```
 select x from thisCollection x;
```

More than one **DB2NT select** statements may appear for complex OO-SQLI queries, for example:

```
ComponentId: 262251
ProcessId: 365
ThreadId: 458
FunctionName: osql::osql_exec_lazy(char*)
ProbeId: 1071
SourceId: 1.16 src/objsvcs/query/impl/osql.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 10/14/98 15:58:05.732631343
UnitOfWork: 29721:wisneski
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage: The function osql::osql_exec_lazy(char*):1071 reported trace message.
ExtendedMessage: Query = "select x from thisCollection x;"
  Plan = "_begin _plan
  _qes_tuple* _q1;
  _begin
  _i1 = _qes_tuple* _SQL DB2NT fksample [: select q1."ID", q1."X",
```

```
  q2."ID", q2."Y" from person1 q1, person2
  q2 where ( q1."ID" = q2."ID") :]'person', 'person';
  _lazy_for _all _q1 _in _i1 _do _begin
  _print _s2 ("1" _make_bo ("thisCollection", _make_do
  ("personDOImplDO_DAO", _make_dao
  ("person1PODAO_Table", "person", ((_qes_tuple*)
  _q1)->_c1(%_integer%), ((_qes_tuple*) _q1)->_c2(%_integer%)),
  _make_dao ("person2PODAO_Table", "person", ((_qes_tuple*)
  _q1)->_c3(%_integer%), ((_qes_tuple*)
  _q1)->_c4(%_integer%)))));
  _end
  _s0 = _all _s2 ;
  _end
  _end _plan
  "
RawDataLen: 0
```

**RELATED REFERENCES**

“Chapter 9. Cache Service Trace” on page 33
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Troubleshooting Cache Related Problems” on page 74

# Chapter 9. Cache Service Trace

The cache service trace records requests to the cache, data exchanges between the transaction cache and the global cache, and SQL operations made to the database. This topic describes how to turn on cache service trace and interpret the trace output:



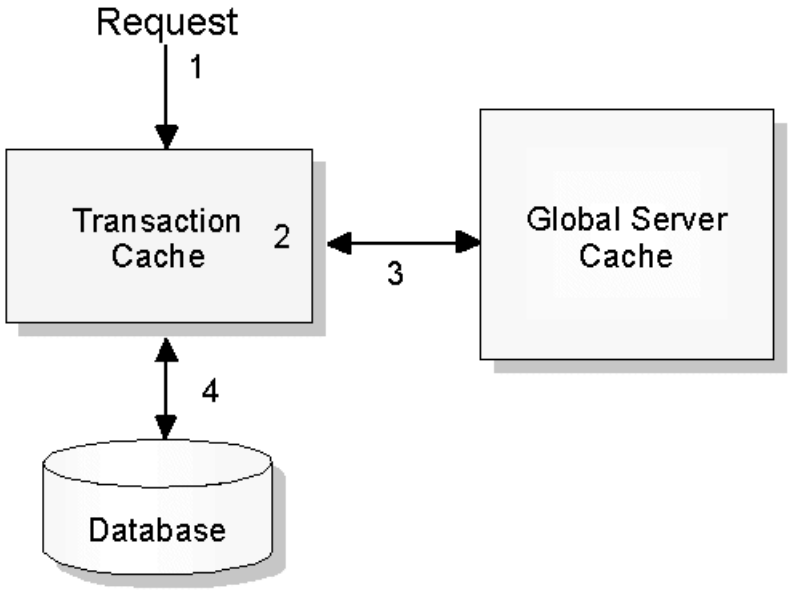
**Setting the Cache ServiceTrace Level**
To set the cache service trace level, complete the following steps:

1. Display the System Manager user interface, and set the user-level to **Expert**.
2. Expand the **Host Images** folder.
3. From the pop-up menu of the Server Image for the application server that you are interested in, click **Edit**. This displays the Object Editor for the Server Image. **Note**: You do not have to stop and start the server to set this trace.
4. In the Object Editor window, click the **Component Trace** tab.
5. Set the **caching component trace level** attribute value to **advanced**, to enable trace information to be recorded.

This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on the formatting of trace logs.

**Interpreting the Cache Service Trace**
To work through cache component trace information, you need to understand the flow of processing within the cache service.

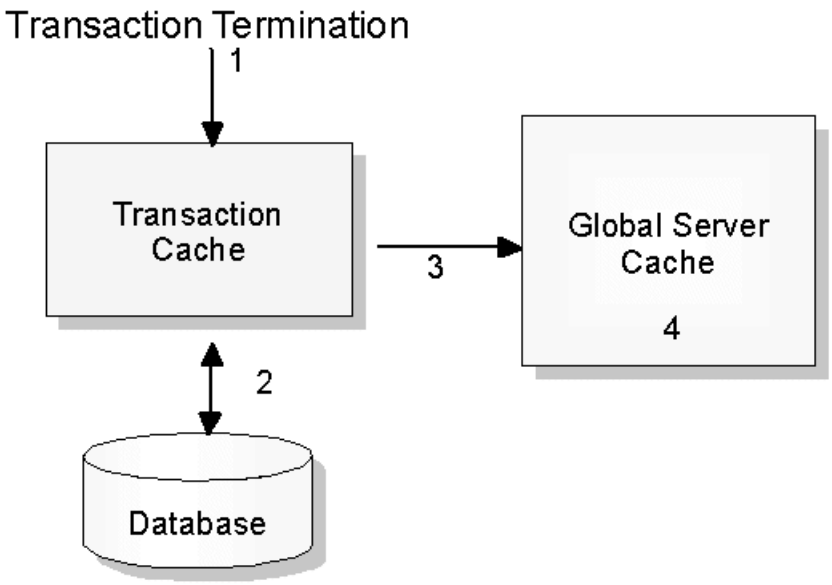
**The Flow of Control for the Cache Service**
The following figure shows the flow within the cache service. Descriptions are provided after the figure.



1. A request for data is made. This typically comes from a DataObject and is made using the DAO (Data Access Object) interface. Often this is a request to read or update data values, or to retrieve, create or delete a database record.
2. For each transaction, the cache service maintains a transaction cache containing data for that transaction. If the data requested is already in the transaction cache, the data is returned or updated. When data is updated, the database may be updated immediately or the update may be deferred until commit is done.
3. If the data is not contained in the transaction cache, and if the ManagedObject was configured for optimistic caching, then a lookup is done on the Global Server Cache. If the data is found there and the data is not older than the time period indicated by the refreshInterval specified by the configuration, the data is copied into the transaction cache and returned to the caller.
4. If the data is found neither in the transaction nor in the global cache, then the data is retrieved from the database. For optimistic caching, the database lock is released after retrieval. For pessimistic caching, the database lock is retained for the duration of the transaction.

**Flow of Control at Transaction Termination**

The following figure shows the flow within the cache service when a transaction terminates. The flow of control is described after the figure.




At transaction termination, if the transaction abended, then the transaction cache is destroyed and any updates made to the database are rolled back. The global cache is not updated. If the transaction was committed then the following events occur:

1. Updated data is written to the database during the beforeCompletion phase of commit processing.
2. During the afterCompletion phase of commit processing, data that is optimistically cached is copied to the global cache, provided the refreshInterval is not zero and the global cache does not already contain a more recent copy of the data.

3. If the global cache becomes full, any data contained in the global cache beyond the refreshInterval is purged and then other data is purged as needed to keep the size of the Global Cache less than the maximum value specified in the System Management Profile object.

### Examples of Cache Service Trace Records

A formatted cache trace file contains entries similar to the following examples.

#### Example 1: Cache Service Trace

The following trace record shows an example of a retrieve request made to the cache service. The mapExpr and keyStr refer to metadata in the Application system management object that indicates the table and primary key of the database record.

```
PrimaryMessage: The function
ICache_RDBDao_Impl::retrieve():987 reported trace data.
ExtendedMessage: mapExpr = "rolePODAO_Table(rolePO_Alias)"
keyStr = "64000000"
```

#### Example 2: Cache Service Trace

The following trace record is an example of a request to retrieve the value of an attribute of a cache entry. The attribute is number 2. The datatype of the attribute is long and the value is 3.

```
ICache_RDBDao_Impl::getValueById(ICache::AttrId,ICache::AttrValue&):482
reported trace message.
ExtendedMessage: value of attrNum:2 = "(long, 3)"
```

#### Example 3: Cache Service Trace

The following trace records show data records being found in the global cache and copied to the transaction cache.

```
PrimaryMessage: The function
  GlobalCacheSpace::address(vtable*,RDBKey*,int):702 reported trace message.
ExtendedMessage: GlobalCache Address: requested entry = "vtbl->id = 2, key =
  05000000"
PrimaryMessage: The function GlobalCacheSpace::address(vtable*,RDBKey*,int):795
  reported trace message.
ExtendedMessage: GlobalCache Address: entry found = "num_entries = 1, ri = 60,
  vtbl->id = 2, key = 05000000, current ts = 1998-09-25-12.36.26.000000, entry
  ts + ri = 1998-09-25-12.37.11.000000"
PrimaryMessage: The function CacheEntry::dump() const:475 reported trace message.
ExtendedMessage: CacheEntry Flags = "nelem = 6, maxElems = 6, isUpdateDefer = 1,
  isLockOptimistic = 1, isNewEntry = 0, isDirty = 0, isDelete = 0, isDBComm = 1,
  isGCMove =0, isWrite = 37411192, ts = 1998-09-25-12.36.11.000000,
  fkAssocsSet = 0, fkUpdated =0"
PrimaryMessage: The function CacheEntry::dump() const:494 reported trace message.
ExtendedMessage: CacheEntry Attributes = "attr[0]: name = ID type = integer
  old_value = 5 old_len = 4
  attr[1]: name = A type = integer old_value = 6 old_len = 4
  attr[2]: name = B type = integer old_value = 6 old_len = 4
  attr[3]: name = C type = integer old_value = NULL old_len = -1
  attr[4]: name = DEPT type = integer old_value = 2 old_len = 4
  attr[5]: name = DTYPE type = string old_value = PERSON old_len = 6
```

#### Example 4: Cache Service Trace

The following trace records show an example of the cache service being called at transaction termination. For commit, there is a beforeCompletion and an afterCompletion call.

```
PrimaryMessage: The function
  ICacheMgr_beforeCompletion(const OMGtid_ptr):121 reported a message.
PrimaryMessage: The function ICacheMgr_afterCompletion(const OMGtid_ptr,const
  CosTransactions::Status):137 reported a message.
```

**Example 5: Cache Service Trace**

The following trace records show an SQL statement being executed against the database, the input and output data variables, and the sqlcode.

```
PrimaryMessage: The function
  db2emb_access::sql_debug():1581 reported trace message.
ExtendedMessage: exec = " prepare stmt :: select q0.ID, q0.A, q0.B, q0.C, q0.DEPT,
  q0.DTYPE from org q0 where q0.ID = ?, cursor: 1"
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
PrimaryMessage: The function db2emb_access::sql_debug():1581 reported trace message.
ExtendedMessage: exec = " sql open cursor: 1"
PrimaryMessage: The function db2emb_access::dump_sqlda(sqlda*):1423 reported trace
  message.
ExtendedMessage: sqldaparm = "
  sqlvar[0](4,0) ID = 4, 05
  "
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
PrimaryMessage: The function db2emb_access::sql_debug():1581 reported trace message.
ExtendedMessage: exec = " sql fetch called, cursor: 1"
PrimaryMessage: The function db2emb_access::dump_sqlda(sqlda*):1419 reported trace
  message.
ExtendedMessage: dbsqlda = "
  sqlvar[0](4,0) ID = 4, 05
  sqlvar[1](4,0) A = 4, 011
  sqlvar[2](4,0) B = 4, 06
  sqlvar[3](4,-1) C = 4, -1 NULL
  sqlvar[4](4,0) DEPT = 4, 02
  sqlvar[5](20,0) DTYPE = 20, 0PERSON
  "
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
PrimaryMessage: The function db2emb_access::sql_debug():1581 reported trace
  message.
ExtendedMessage: exec = " sql close cursor: 1"
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
```

**Example 6: Cache Service Trace**

The following series of trace records show an update to the database being performed. **lockmode=1** indicates pessimistic cache option (lockmode=2 indicates optimistic). The contents of the SQL variables are dumped. Column A has value 12 and column ID has value 5. The **update** statement is prepared and executed, and the sqlcode is zero.

```
PrimaryMessage: The function
  db2emb_access::db2emb_access
  (vtable*,RDBaccess::_SQL_operator_type,aPolicy*,RDBKey*,FKRel*):296
  reported trace message.
ExtendedMessage: lockmode = 1
  accessmode = 2
  sql_operation = 3
  dbname = "fksample"
PrimaryMessage: The function db2emb_access::dump_sqlda(sqlda*):1419 reported trace
  message.
ExtendedMessage: dbsqlda = "
  sqlvar[0](4,0) A = 4, 012
  sqlvar[1](4,0) ID = 4, 05
  "
PrimaryMessage: The function db2emb_access::sql_debug():1581 reported trace message.
ExtendedMessage: exec = " prepare stmt :: update org q0 set A = ? where q0.ID = ?,
```

```
  cursor: 65"
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
PrimaryMessage: The function db2emb_access::sql_debug():1581 reported trace message.
ExtendedMessage: exec = " sql execute called"
PrimaryMessage: The function db2emb_access::sql_exec(RDBaccess::eExecSql):487
  reported trace message.
ExtendedMessage: sqlcode = 0
```

### Example Seven: Cache Service Trace

These records show data records being copied into the global cache.

```
PrimaryMessage: The function
  GlobalCacheSpace::replace(vtable*,CacheEntry*,RDBKey*):509 reported trace message.
ExtendedMessage: GlobalCache Replace: entry sizes are equal, hence original entry is
  replaced and the new entry = ""
PrimaryMessage: The function CacheEntry::dump() const:475 reported trace message.
ExtendedMessage: CacheEntry Flags = "nelem = 6, maxElems = 6, isUpdateDefer = 1,
  isLockOptimistic = 1, isNewEntry = 0, isDirty = 0, isDelete = 0, isDBComm = 1,
  isGCMove =0, isWrite = 39843584, ts = 1998-09-25-12.36.11.000000,
  fkAssocsSet = 0, fkUpdated =0"
PrimaryMessage: The function CacheEntry::dump() const:494 reported trace message.
ExtendedMessage: CacheEntry Attributes = "attr[0]: name = ID type = integer
  old_value = 5 old_len = 4
  attr[1]: name = A type = integer old_value = 7 old_len = 4
  attr[2]: name = B type = integer old_value = 6 old_len = 4
  attr[3]: name = C type = integer old_value = NULL old_len = -1
  attr[4]: name = DEPT type = integer old_value = 2 old_len = 4
  attr[5]: name = DTYPE type = string old_value = PERSON old_len = 6
```

#### RELATED CONCEPTS

Data Cache Considerations (*System Administration Guide*)

#### RELATED REFERENCES

“Chapter 8. Query Service Trace” on page 27
“Troubleshooting Cache Related Problems” on page 74
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72

#### RELATED TASKS

Configure DB2 to use the CBConnector Data Cache (*System Administration Guide*)
Configure a Cache for SQL data (*System Administration Guide*)

# Chapter 10. ORB Request Trace

ORB request trace can be enabled on the server to display the target object type/class and the function name. ORB request tracing is a fast way to let you view function call flow and understand how objects work with each other. After doing an ORB request trace, if you still require more information to help you diagnose the problem, you can run the XYZ trace that is described in “Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace” on page 51. The XYZ trace shows the Managed Object Framework (MOFW) function calls. If more details, such as input and output parameters, are required then ORB communication tracing may be more appropriate.

This topic describes how to turn on the ORB request trace and interpret the trace output:



### Setting the ORB Request Trace

To set the ORB request trace, perform these steps:

1. Display the System Manager user interface, and set the user level to **Expert**.
2. Expand your **Host Images**.
3. Expand the **Server Images** folder.
4. Left click on your server image to see if it is running. The status bar at the bottom of the System Manager user interface application displays the state (and health) of the selected server. **Note**: You do not have to stop the server to set this trace.
5. Right click on your server image and select **Edit**. This displays the Object Editor for the Server Image.
6. In the Object Editor window, click the **Component Trace** tab.
7. Set the **ORB request trace level** attribute value to **Advanced**. Click **Apply** and then **OK** to enable the trace.

This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on the formatting of trace logs.

### Sample: Formatted ORB Request Trace Output

The formatted output file looks like the formatted activity log. Function calls are recorded in the activity log. The function name can be seen in the functionName or PrimaryMessage fields. Here is an example of the contents of a formatted output file:

```
_____________________________________-
ComponentId: 393319
ProcessId: 567
ThreadId: 534
FunctionName: CORBA::BOA::local_object_to_object_key(CORBA::Object_ORBProxy_ptr)
ProbeId: 2990
SourceId: 1.66 src/orb/src/somd/boa.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
```

```
clientUserId:
TimeStamp: 10/6/98 9:54:14.343609431
UnitOfWork:
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage: The function
  CORBA::BOA::local_object_to_object_key(CORBA::Object_ORBProxy_ptr):2990
  reported data.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 393319
ProcessId: 567
ThreadId: 534
FunctionName: CORBA::Request::send_deferred()
ProbeId: 1405
SourceId: 1.57.1.2 src/orb/src/request/request.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
clientUserId:
TimeStamp: 10/6/98 9:54:14.371803789
UnitOfWork:
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage: The function CORBA::Request::send_deferred():1405 reported data.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 393319
ProcessId: 567
ThreadId: 534
FunctionName: CORBA::Request::invoke()
ProbeId: 1338
SourceId: 1.57.1.2 src/orb/src/request/request.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
clientUserId:
TimeStamp: 10/6/98 9:54:16.488164076
UnitOfWork:
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage: The function CORBA::Request::invoke():1338 reported data.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
.
.
.
```

**RELATED REFERENCES**

“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Chapter 2. Activity Log for Problem Determination” on page 3
“Chapter 11. ORB Communication Trace” on page 41

# Chapter 11. ORB Communication Trace

ORB communication trace provides ORB packet dumps. If you want to do low level debugging and look at ORB packets, set the ORB communication trace to capture this information.

This topic describes how to turn on the ORB communication trace and interpret the trace output (GIOP packets):



### Setting the ORB Communication Trace

To set the ORB communication trace, perform these steps:

1. Display the System Manager user interface, and set the user level to **Expert**.
2. Expand your **Host Images**.
3. Expand the **Server Images** folder
4. Left click on your server image to see if it is running. The status bar at the bottom of the System Manager user interface application displays the state (and health) of the selected server. **Note**: You do not have to stop the server set this trace.
5. Right click on your server image and select **Edit**. This displays the Object Editor for the Server Image.
6. In the Object Editor window, click the **Component Trace** tab.
7. Set the **ORB communication trace level** attribute value to **Advanced**. Click **Apply** and then **OK** to enable the trace.

This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on the formatting of trace logs.

### Sample ORB Communication Trace

The formatted output file looks like the formatted activity log. The ORB packet dumps are shown as **RawData**, as follows:

```
.
.
.
-------------------------------------
ComponentId: 393317
ProcessId: 647
ThreadId: 157
FunctionName: IPTransport::send_message(Encapsulation*)
ProbeId: 1135
SourceId: 1.26 src/orb/src/trans/transip.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
clientUserId:
TimeStamp: 10/6/98 10:03:57.132667914
UnitOfWork:
Severity: 3
Category: 3
```

```
FormatWarning: 0
PrimaryMessage:
The function IPTransport::send_message(Encapsulation*):1135 reported trace message.
ExtendedMessage:
RawDataLen: 57
RawData:
0000  47 49 4F 50 01 01 01 03 - 2D 00 00 00 02 00 00 00  GIOP....-.......
0010  25 00 00 00 2F 2F 4C 49 - 42 4D 44 3A 2F 2F 4C 49  %.../LIBMD://LI
0020  42 4D 44 3A 2F 2F 4C 49 - 42 4D 44 3A 2F 2F 4C 49  BMD://LIBMD://LI
0030  42 4D 44 3A 46 41 4B 45 - 00                       BMD:FAKE.
-----------------------------------
ComponentId: 393317
ProcessId: 647
ThreadId: 524
FunctionName: IPTransportHandler::handle_event(void*)
ProbeId: 775
SourceId: 1.26 src/orb/src/trans/transip.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
clientUserId:
TimeStamp: 10/6/98 10:03:57.134744714
UnitOfWork:
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage:
The function IPTransportHandler::handle_event(void*):775 reported trace data.
ExtendedMessage: tstring = "
Message arrived Src:9.53.167.28/900 Dst:babe.austin.ibm.com/3734"
RawDataLen: 0
-----------------------------------
ComponentId: 393317
ProcessId: 647
ThreadId: 524
FunctionName: IPTransportHandler::handle_event(void*)
ProbeId: 872
SourceId: 1.26 src/orb/src/trans/transip.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: PersonServer
clientHostName:
clientUserId:
TimeStamp: 10/6/98 10:03:57.135980904
UnitOfWork:
Severity: 3
Category: 3
FormatWarning: 0
PrimaryMessage:
The function IPTransportHandler::handle_event(void*):872 reported trace message.
ExtendedMessage:
RawDataLen: 20
RawData:
0000  47 49 4F 50 01 01 01 04 - 08 00 00 00 02 00 00 00  GIOP............
0010  01 00 00 00
-----------------------------------
.
.
.                                      ....
```

**RELATED REFERENCES**

"Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace" on page 51

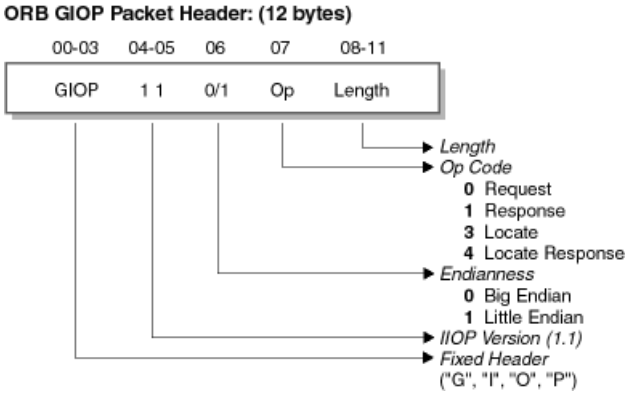
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Chapter 2. Activity Log for Problem Determination” on page 3

## General Inter-ORB Protocol (GIOP) Packets

To interpret the dump in the ORB communications trace, you have to understand the format of the data in the ORB GIOP packets. This section describes the format of the GIOP packet:



**GIOP Packet Header (00-11)**
Each packet begins with a twelve-byte header (bytes 00 through 11). The header identifies the type of General Inter-ORB Protocol (GIOP) packet (which is the Op Code in byte 07) that is being sent and the length of the packet. The most important piece of information in the header is the Op Code which tells you how to interpret the rest of the information in the packet. Here is the layout of the header or first twelve bytes of the GIOP packet:




**Op Code (07)**
There are four possible Op codes: 0, 1, 3, and 4. Each Op code has a different layout of information in the rest of the packet.

**Op Code 0 - Request**
When the Op Code is 0 (Request), this is typically a method invocation. Starting with the thirteenth byte, interpret the packet as follows:

1. Sequence Service Contexts (for RAS, Security, Transaction)
   - Service Tag Sequence (long, 4 bytes)
     Specifies the number of Service Tags.

- Service Tag (long, 4 bytes)
- Service Data:
  - Length (long, 4 bytes)
  - Data (octets specified by Length)

2. Request-ID (long, 4 bytes)
3. Response Expected (Boolean, sent as octet)
   - **0** No response expected (one way method)
   - **1** Response expected
4. Object Key
   - Object Key Length (long, 4 bytes). Start at the next 4-byte boundary.
   - Object Key Data (octets specified by Object Key Length)
5. Method Name (string)
   - String Length (long, 4 bytes)
   - String Data (specified by String Length)
     The NULL terminator is included.
6. Principal
   - Principal Length (long, 4 bytes)
   - Principal Data (octets specified by Principal Length)
7. Method parameters (See .idl file.)
   The method parameters follow the principal data and ends with the last byte in the packet.

See ORB communication trace example 2 (page 47) for more information on how to interpret an Op code 0 GIOP packet.

**Op Code 1 - Response**
When the Op Code is 1 (Response), this is typically a response to a method invocation. Starting with the thirteenth byte, interpret the packet as follows:

1. Sequence Service Contexts (for RAS, Security, Transaction)
   - Service Sequence (long, 4 bytes)
     The number of Service Tags.
     - Service Tag (long, 4 bytes)
     - Service Data:
       - Length (long, 4 bytes)
       - Data (octets specified by Length)
2. Request ID (long, 4 bytes)
   This matches the Request ID of an Op Code 0 request.
3. Status (long, 4 bytes)
   - **0** No Exception
     RETURNVALUE/INOUT/OUT Parameters follows Status.
   - **1** User Exception
     Exception follows Status.
   - **2** System Exception
     Exception follows Status.
   - **3** Forward
     IOR follows Status.
4. Depending on the Status, one of the following items appears after the Status information:

- RETURNVALUE/INOUT/OUT Parameters (See .idl file.)
  The return value can also be an object which is returned as an IOR.
- Exception
  - String Length (long, 4 bytes)
  - String Data (specified by String Length)
    This is the exception. The NULL terminator is included.
- Interoperable Object Reference (IOR)
  Depending on the IIOP version, the IOR can have slightly different formats. Refer to the CORBA: Architecture and Specification for more information.
  - Type ID (string)
    - String Length (long, 4 bytes)
    - String Data (specified by String Length)
      The NULL terminator is included.
  - Profile Sequence Length (long, 4 bytes)
    Specifies the number of Profile Tags.
    - Profile Tag
      Each Profile Tag consists of the following information:
      - Tag (4 bytes)
        - **0** TCP/IP
      - Profile Length (long, 4 bytes)
      - Profile Data (specified by Profile Length)
        - Version (2 bytes)
        - TCP/IP Address String
          - String Length (long, 4 bytes)
          - String Data (specified by String Length)
        - Port (short, 2 bytes)
          It is the TCP/IP listening port.
        - Object Key
          - Object Key Length (long, 4 bytes)
          - Object Key (octets specified by Object Key Length)
        - Component Tag Sequence
          Specifies the number of Component Tags.
          - Component
            Each Component Tag consists of the following information:
            - Tag (4 bytes)
            - Component Data Length (long, 4 bytes)
            - Component Data (specified by Component Length)

See ORB Communications Trace example 3 (page 48) for more information on how to interpret an Op code 1 GIOP packet.

**Op Code 3 - Locate**

When the Op Code is 3 (Locate), this is a request to locate an object. Starting with the thirteenth byte, interpret the packet as follows:

1. Request ID (long, 4 bytes)
   This is the Packet Serial Number (correlation ID).
2. Object Key
   - Object Key Length (long, 4 bytes). Start at the next 4-byte boundary.

- Object Key Data (octets specified by Object Key Length)

See ORB Communications Trace example 1 (page 46) for more information on how to interpret an Op code 3 GIOP packet.

**Op Code 4 - Locate Response**
When the Op Code is 4 (Locate Response), this is a response to a Locate Object request. Starting with the thirteenth byte, interpret the packet as follows:

1. Request ID (long, 4 bytes)
   This matches the Request ID of the Locate request.
2. Response Code (long, 4 bytes)
   - **0** Unknown object
   - **1** Object Here
   - **2** Object Forward
     New IOR follows.
3. Interoperable Object Reference (IOR)
   Depending on the IIOP version, the IOR can have slightly different formats. Refer to the CORBA: Architecture and Specification for more information.
   - Type ID (string)
     – String Length (long, 4 bytes)
     – String Data (specified by String Length)
       The NULL terminator is included.
   - Profile Sequence Length (long, 4 bytes)
     Specifies the number of Profile Tags.
     – Profile Tag
       Each Profile Tag consists of the following info.
       - Tag (4 bytes)
         - **0** TCP/IP
       - Profile Length (long, 4 bytes)
       - Profile Data (specified by Profile Length)
         - Version (2 bytes)
         - TCP/IP Address String
           – String Length (long, 4 bytes)
           – String Data (specified by String Length)
         - Port (short, 2 bytes)
           It is the TCP/IP listening port.
         - Object Key
           – Object Key Length (long, 4 bytes)
           – Object Key (octets specified by Object Key Length)
         - Component Tag Sequence
           Specifies the number of Component Tags.
           – Component
             Each Component Tag consists of the following info.
             - Tag (4 bytes)
             - Component Data Length (long, 4 bytes)
             - Component Data (specified by Component Length)

**Example 1: ORB Communications Trace (Op Code 3)**

```
0000    47 49 4f 50 01 01 01 03 b0 00 00 00 04 00 00 00     GIOP....ª.......
0010    a8 00 00 00 04 32 33 34 38 36 41 41 31 2d 44 32     ¬....23486AA1-D2
0020    33 32 2d 31 35 45 36 2d 46 31 45 33 2d 30 32 31     32-15E6-F1E3-021
0030    32 30 39 33 35 41 36 33 41 00 02 00 00 52 03 00     20935A63A....R..
0040    00 35 00 0f 00 69 42 4f 49 4d 43 6f 6e 74 61 69     .5...iBOIMContai
0050    6e 65 72 00 03 00 10 00 69 43 44 53 43 61 63 68     ner.....iCDSCach
0060    65 64 48 6f 6d 65 73 00 0e 00 53 6f 6d 43 6f 6e     edHomes...SomCon
0070    74 61 69 6e 65 72 49 00 42 00 14 00 69 42 4f 49     tainerI.B...iBOI
0080    4d 48 6f 6d 65 4f 66 52 65 67 48 6f 6d 65 73 00     MHomeOfRegHomes.
0090    03 00 1d 00 43 6c 61 69 6d 41 70 70 5f 43 6c 61     ....ClaimApp_Cla
00a0    69 6d 4d 4f 5f 43 6c 61 69 6d 44 4f 49 6d 70 6c     imMO_ClaimDOImpl
00b0    00 09 00 53 6f 6d 48 6f 6d 65 49 00                 ...SomHomeI.
```

Interpret this ORB communication trace example as follows:

1. Find out the Op code.
   The header, 47 49 4f 50 01 01 01 **03** b0 00 00 00, shows "03" Locate for Op code. Use the Op code 3 pattern to interpret the subsequent information in the dump.
2. Packet serial number is 04 00 00 00
3. Object Key Length is a8 00 00 00 (hex a8 is 168 in decimal).
4. Object Key is the next 168 bytes.

**Example 2: ORB Communications Trace (Op Code 0)**

```
0000    47 49 4f 50 01 01 01 00 a3 01 00 00 01 00 00 00     GIOP....·.......
0010    02 4d 42 49 28 00 00 00 01 00 00 00 1b 00 00 00     .MBI(...........
0020    39 31 34 36 3a 61 73 64 61 73 64 2e 61 75 73 74     9146:asdasd.aust
0030    69 6e 2e 69 62 6d 2e 63 6f 6d 00 00 00 00 00 00     in.ibm.com......
0040    05 00 00 00 01 00 00 00 b7 00 00 00 04 33 42 37     .........+....3B7
0050    36 32 41 39 34 2d 44 32 32 42 2d 31 35 45 36 2d     62A94-D22B-15E6-
0060    46 31 45 33 2d 30 32 31 32 30 39 33 35 41 36 33     F1E3-02120935A63
0070    41 00 02 00 00 52 03 00 00 33 00 0f 00 69 42 4f     A....R...3...iBO
0080    49 4d 43 6f 6e 74 61 69 6e 65 72 00 03 00 0e 00     IMContainer.....
0090    69 54 72 61 6e 73 53 79 73 4f 62 6a 73 00 0e 00     iTransSysObjs...
00a0    53 6f 6d 43 6f 6e 74 61 69 6e 65 72 49 00 53 00     SomContainerI.S.
00b0    16 00 69 43 44 53 4e 61 6d 69 6e 67 43 6f 6e 74     ..iCDSNamingCont
00c0    65 78 74 48 6f 6d 65 00 39 00 2f 2e 3a 2f 43 42     extHome.9./.:/CB
00d0    43 2d 6c 6f 63 61 6c 2d 72 6f 6f 74 73 2f 31 41     C-local-roots/1A
00e0    32 38 32 34 33 32 2d 44 33 34 39 2d 31 35 45 36     282432-D349-15E6
00f0    2d 45 37 45 35 2d 30 32 30 38 30 39 33 35 41 36     -E7E5-02080935A6
0100    33 41 00 00 14 00 00 00 72 65 73 6f 6c 76 65 5f     3A......resolve_
0110    77 69 74 68 5f 73 74 72 69 6e 67 00 22 00 00 00     with_string."...
0120    01 49 42 4d 44 3a 00 00 16 00 00 00 61 73 64 61     .IBMD:......asda
0130    73 64 2e 61 75 73 74 69 6e 2e 69 62 6d 2e 63 6f     sd.austin.ibm.co
0140    6d 00 00 00 67 00 00 00 68 6f 73 74 2f 72 65 73     m...g...host/res
0150    6f 75 72 63 65 73 2f 66 61 63 74 6f 72 69 65 73     ources/factories
0160    2f 73 6f 6d 6c 63 52 65 70 6f 73 69 74 6f 72 79     /somlcRepository
0170    2f 66 61 63 74 6f 72 79 42 72 61 6e 63 68 2f 43     /factoryBranch/C
0180    6c 61 69 6d 2f 4d 79 53 65 72 76 65 72 2a 63 6f     laim/MyServer*co
0190    6e 74 61 69 6e 65 72 4f 66 43 6c 61 69 6d 73 2a     ntainerOfClaims*
01a0    43 6c 61 69 6d 4d 4f 46 61 63 74 6f 72 79 00        ClaimMOFactory.
```

Interpret this ORB communication trace example as follows:

1. Find out the Op code.
   The header, 47 49 4f 50 01 01 01 **00** a3 01 00 00, shows "00" Request for Op code. Use the Op code 0 pattern to interpret the subsequent information in the dump.
2. Sequence Service Contexts:
   a. Sequence (long) is 01 00 00 00 so there is one set of Service Tags.
   b. Service Tag
      - Tag (long) is 02 4d 42 49 28.

- Service Data: Length (long) is 28 00 00 00 (hex 28 is 40 in decimal), and data is the next 40 bytes.

3. Request ID (long) is 05 00 00 00.
4. Request Expected is 01 which is "Response expected".
5. Start at the next 4-byte boundary for "long". Object key length is long (4 bytes). The length is b7 00 00 00 (hex b7 is 183 in decimal). The object key data is in the next 183 bytes.
6. Method Name:
   a. After lining up to next four bytes, method name's length is 14 00 00 00 (hex 14 is 20 in decimal).
   b. The method name is the next 20 bytes which is 72 65 .... 69 6e 67 00 (resolve_with_string.).
7. Principal:
   a. Principal length (long) is 22 00 00 00 (hex 22 is 34 in decimal).
   b. Principal data is the next 34 bytes.
8. After lining up to next four bytes, method parameters start with 64 00 00 00 ... and continues to the end of the dump (g...../Claim/MyServer*containerOfClaims*ClaimMOFactory.).

**Example 3: ORB Communications Trace (Op Code 1)**

```
0000  47 49 4F 50 01 01 01 01 - 78 01 00 00 00 00 00 00  GIOP....x.......
0010  EB 00 00 00 00 00 00 00 - 35 00 00 00 49 44 4C 3A  ........5...IDL:
0020  49 42 4F 49 4D 45 78 74 - 4D 61 6E 61 67 65 64 4F  IBOIMExtManagedO
0030  62 6A 65 63 74 2F 49 51 - 75 65 72 79 61 62 6C 65  bject/IQueryable
0040  49 74 65 72 61 62 6C 65 - 48 6F 6D 65 3A 31 2E 30  IterableHome:1.0
0050  00 00 00 00 01 00 00 00 - 00 00 00 00 24 01 00 00  ............$...
0060  01 01 01 00 0C 00 00 00 - 39 2E 35 33 2E 31 36 37  ........9.53.167
0070  2E 32 38 00 95 0E 00 00 - C9 00 00 00 04 32 43 46  .28..........2CF
0080  31 43 42 42 37 2D 43 45 - 44 41 2D 31 36 31 38 2D  1CBB7-CEDA-1618-
0090  45 41 45 32 2D 30 30 35 - 44 30 39 33 35 41 37 31  EAE2-005D0935A71
00A0  43 00 02 00 00 52 03 00 - 00 35 00 0F 00 69 42 4F  C....R...5...iBO
00B0  49 4D 43 6F 6E 74 61 69 - 6E 65 72 00 03 00 10 00  IMContainer.....
00C0  69 43 44 53 43 61 63 68 - 65 64 48 6F 6D 65 73 00  iCDSCachedHomes.
00D0  0E 00 53 6F 6D 43 6F 6E - 74 61 69 6E 65 72 49 00  ..SomContainerI.
00E0  63 00 16 00 69 42 4F 49 - 4D 48 6F 6D 65 4F 66 52  c...iBOIMHomeOfR
00F0  65 67 51 49 48 6F 6D 65 - 73 00 03 00 3C 00 50 65  egQIHomes...<.Pe
0100  72 73 6F 6E 41 70 70 5F - 50 65 72 73 6F 6E 4D 6F  rsonApp_PersonMo
0110  64 4D 4F 5F 50 65 72 73 - 6F 6E 4D 4F 5F 50 65 72  dMO_PersonMO_Per
0120  73 6F 6E 4D 6F 64 44 4F - 49 6D 70 6C 5F 50 65 72  sonModDOImpl_Per
0130  73 6F 6E 44 4F 49 6D 70 - 6C 00 09 00 53 6F 6D 48  sonDOImpl...SomH
0140  6F 6D 65 49 00 00 00 00 - 02 00 00 00 09 03 00 00  omeI............
0150  1D 00 00 00 01 0C 00 00 - 00 50 65 72 73 6F 6E 53  .........PersonS
0160  65 72 76 65 72 47 53 53 - 5F 44 43 45 00 43 00 6D  erverGSS_DCE.C.m
0170  69 00 00 00 14 00 00 00 - 08 00 00 00 01 00 7E 00  i.............~.
0180  7E 00 BC 0B                                        ~...
```

The above example is a response to a request to find a factory of persons. Interpret this ORB communication trace example as follows:

1. Find out the Op code.
   The header, 47 49 4F 50 01 01 01 **01** 78 01 00 00, shows "01" Request for Op code. Use the Op code 1 pattern to interpret the subsequent information in the dump. Note the Little Endian setting (01) in the seventh byte.
2. Sequence Service Contexts:
   a. Length of Sequence (4 bytes) is 00 00 00 00. There are no Sequence Service Tags.
3. Request-ID is EB 00 00 00.

4. Status is 00 00 00 00 (No Exception).

   When there is no exception, the return value of the method can be an object, as shown in this example. The object is returned as an IOR, and the sender of the request converts this IOR into an object.

5. Interoperable Object Reference (IOR)
   - Type ID
     – String Length (4 bytes) is 35 (which is 53 in decimal).
     – String Data is 49 44 4C 3A...3A 31 3E 30 00 (IDL:..:1.0.).
   - Profile Tag Sequence
     Start at the next 4-byte boundary, the sequence count is 01 00 00 00 so there is only one set of Profile Tags.
     – Profile Tag contains the following:
       - Tag (4 bytes) is 00 00 00 00 which is a TCP/IP profile.
       - Profile Length (4 bytes) is 24 01 00 00 (Little Endian).
         There are 124 (or 292 in decimal) bytes of profile data which takes you to the end of the dump.
       - Profile Data
         • Version (2 bytes) is 01 01.
         • TCP/IP Address String
           – String Length (4 bytes) is 0C 00 00 00 which is 12 bytes.
           – String Data is 39 2E...2E 32 38 00 (9.53.167.28.).
         • TCP/IP listening port (2 bytes) is 95 0E.
         • Object Key
           – Start at the next 4-byte boundary, Object Key Length (4 bytes) is C9 00 00 00 (or 201 in decimal).
           – Object Key is 04 32 43 46... 6F 6D  65 49 00.
         • Component Tags
           This is a 1.1 style IOR which contains a Sequence of tagged components after the object key.
           – Component Tag Sequence
             Start at the next 4-byte boundary, the Sequence count is 02 00 00 00 so there are two sets of Component Tags.
             - First Component
               • Tag (4 bytes) is 09 03 00 00.
               • Component Tag Length (4 bytes) is 1D 00 00 00.
                 There are 1D (or 29 in decimal) bytes of Component data.
               • Component data is 01 0C 00 00 ...00 43 00 6D 69.
             - Second Component
               • Align to the next 4 bytes, Tag (4 bytes) is 14 00 00 00.
               • Component Tag Length (4 bytes) is 08 00 00 00.
                 There are 8 bytes of Component data.
               • Component Data is 01 00 7E 00 7E 00 BC 0B (end of the packet).

**RELATED REFERENCES**

“Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace” on page 51
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Chapter 2. Activity Log for Problem Determination” on page 3

# Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace

When debugging your applications, you may want to trace the sequence of function or method calls when the application is running. One way to do this is to put print statements (**cout** for C++ or **System.out.println** for Java) in your user code.

This topic describes how to trace Managed Object Framework (MOFW) function calls:



### Tracing MOFW Function Calls

If you want to trace the set of framework function calls in the generated code, you have to put **cout** statements in Object Builder's generated .cpp files. The following steps describe how you can create a macro, XYZ, to perform function call tracing, and direct the console log output to a file to view the results.

1. After you have the generated code from the Object Builder, add the following statements to all *_I.cpp and *PO.cpp files.
   a. **#define XYZ**
      Define a macro, XYZ, for the **cout** statement. For example, in PolicyBO_I.cpp file, add the #define XYZ line as follows:

```
// Generated from file PolicyBO.idl
// on Friday, July 24, 1998 9:30:36 o'clock AM CDT 01290636449
// by Object Builder
//Version identifier DCE:99FE00AB-DEC6-11d1-B431-08005ACE0236:1
#ifdef SOMCBNOLOCALINCLUDES
#include <PolicyBO.ih>
#include <PolicyKey.hh>
#else
#include "PolicyBO.ih"
#include "PolicyKey.hh"
#endif
// Generated from interface PolicyBO
// Version identifier DCE:99FE00AC-DEC6-11d1-B431-08005ACE0236:1
#define XYZ cout<<__FILE__<<":"<<__LINE__<<":"<<__FUNCTION__<<endl
PolicyBO_Impl::PolicyBO_Impl()
.
.
.
```

   b. **XYZ macro calls**
      For each function in each *_I.cpp and *PO.cpp file, add the XYZ macro call. For example, in PolicyBO_I.cpp file add the macro calls as follows:

```
.
.
.
PolicyBO_Impl::PolicyBO_Impl()
{XYZ; }
::CORBA::Float PolicyBO_Impl::amount()
{XYZ;
  //Version identifier DCE:99FE008D-DEC6-11d1-B431-08005ACE0236:1
  // Insert Method modifications here
  return iDataObject->amount();
  // End Method modifications here
}
::CORBA::Void PolicyBO_Impl::amount( ::CORBA::Float amount)
{XYZ;
```

```
  //Version identifier DCE:99FE008D-DEC6-11d1-B431-08005ACE0236:2
  // Insert Method modifications here
  iDataObject->amount(amount);
  // End Method modifications here
}
::CORBA::Long PolicyBO_Impl::policyNo()
{XYZ;
  //Version identifier DCE:99FE008E-DEC6-11d1-B431-08005ACE0236:1
  // Insert Method modifications here
  return iDataObject->policyNo();
  // End Method modifications here
}
.
.
.
::CORBA::Void PolicyBO_Impl::internalize_from_stream(::CosStream::StreamIO_...
{XYZ;
  //Version identifier DCE:C0AEDB6E-E10A-11d1-B432-08005ACE0236:1
  // Insert Method modifications here
  iDataObject->amount(sourceStreamIO->read_float());
  iDataObject->policyNo(sourceStreamIO->read_long());
  iDataObject->premium(sourceStreamIO->read_float());
  // End Method modifications here
}
::ByteString* PolicyBO_Impl::getPrimaryKeyString()
{XYZ;
  //Version identifier DCE:C0AEDB6F-E10A-11d1-B432-08005ACE0236:1
  // Insert Method modifications here
  PolicyKey_var policyKey = PolicyKey::_create();
  policyKey->policyNo(iDataObject->policyNo());
  return policyKey->toString();
  // End Method modifications here
}
```

2. Build your application.
3. Direct the console log to a file for the run time.
   a. In the System Manager user interface, under server images, right click on the server image to be modified.
   b. Select **Edit**.
   c. Tab to **Log Controls**.
   d. Select **Console Disposition** to file.
   e. Change the name of the file if you want. It will go to a directory named %SOMCBASE%\service\server*servername*. Even though each server has its own directory, name the file, for example, *servername.*console.log, so that later, when all consoles are gathered into a specific place, consoles from various servers will be distinguishable.
   f. After editing the server image, you must restart the server. Right click on the server image and select **Run immediate**.
4. Run the application and look at the function call sequence in the *servername.*console.log.

**Sample XYZ Trace Output in the Console Log**

After implementing the **cout** tracing with the XYZ macro, you will get a console.log file that looks like the following:

```
Server PolicyServer is starting
SOMInit started !!!!!!!!!!!!!!!
Server path = 0HostISomServIPolicyServer
Objects awaiting disposal:
Objects non-smo dead:
Number of elements awaiting disposal or dead non SMOs = 0
SOMInit finished
```

```
   IVB3639I: Remote debug is not enabled for server PolicyServer.
Server PolicyServer is ready
PolicyHomeMO_I.cpp:38:PolicyHomeMO_create()
PolicyHomeBO_I.cpp:24:PolicyHomeBO_Impl::PolicyHomeBO_Impl()
PolicyHomeMO_I.cpp:45:PolicyHomeMO_Impl::PolicyHomeMO_Impl()
PolicyHomeMO_I.cpp:305:PolicyHomeMO_Impl::_incref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
PolicyHomeMO_I.cpp:312:PolicyHomeMO_Impl::_decref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
PolicyHomeMO_I.cpp:38:PolicyHomeMO_create()
PolicyHomeBO_I.cpp:24:PolicyHomeBO_Impl::PolicyHomeBO_Impl()
PolicyHomeMO_I.cpp:45:PolicyHomeMO_Impl::PolicyHomeMO_Impl()
PolicyHomeMO_I.cpp:305:PolicyHomeMO_Impl::_incref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
PolicyHomeMO_I.cpp:312:PolicyHomeMO_Impl::_decref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
PolicyHomeMO_I.cpp:305:PolicyHomeMO_Impl::_incref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
PolicyHomeMO_I.cpp:305:PolicyHomeMO_Impl::_incref()
PolicyHomeMO_I.cpp:31:PolicyHomeMO_Impl::getMixin()
.
.
.
```

**RELATED REFERENCES**

“Merge cout Trace and ORB Communication Trace” on page 104
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Chapter 11. ORB Communication Trace” on page 41

# Chapter 13. Security Service Trace

The security service trace can be used to display information about the flow of control for the security service on a client or application server. This topic describes how to turn on the security service trace and interpret the trace output:

- Setting Security Service Trace (page 55)
- Security Trace Examples (page 57)

**Setting Security Service Trace**

To trace the security flow of control, you can set the object service security trace level in any of the following ways.

- To start temporary tracing of a specific application server or client style (page 55), change the security trace level attribute of the Server Image or Client Style through the System Manager user interface. Tracing will continue at the set level until either the level is changed or the system management Configuration that defines the application server or client style is activated again.
- To start more long-term tracing of application servers or client styles (page 56), change the security trace level attribute of the Server Group, Server (freestanding), or Client Style in a system management Configuration through the System Manager user interface. Tracing will continue at the level set until either the Configuration is changed and activated again or the level is changed on specific Server Images or Client Style Images.
- To start tracing of all application servers and client styles on a host (page 57), change the SecurityTraceLevel environment variable on that host. Tracing will continue at the level set until the level is changed on the environment variable or through the System Manager user interface (as above).

**Notes:**

1. If the trace level for an application server or client style is set through both the SecurityTraceLevel environment variable and the security trace level attribute, the highest value of the two levels is used by the security service.
2. The output of the trace log entries can be found in the activity.log file which is in the `Cbroker\service` directory.
3. For examples of security trace log entries, see Security Trace Examples (page 57).

**Setting the Security Service Trace for Temporary Tracing**

To set the security trace level for temporary tracing of an individual application server or client style, use the System Manager user interface to complete the following steps:

1. Display the System Manager user interface, and set the user-level to **Expert**.
2. Expand the **Host Images** folder.
3. Expand the Host Image on which the application server or client style is running.
4. Set the Security trace level attribute.
   - To set the trace level for an application server, complete the following steps:
     a. Expand the **Server Images** folder.
     b. From the pop-up menu of the Server Image for the application server, click **Edit**. This displays the Object Editor for the Server Image.

- To set the trace level for a client style, complete the following steps:
  a. Expand the **Client Style Images** folder.
  b. From the pop-up menu of the Client Style Image for the client style, click **Edit**. This displays the Object Editor for the Client Style.

5. In the Object Editor window, click the **Object Services Trace** tab.
6. Set the **security trace level** attribute value to **basic** or **intermediate**, to enable the following trace information to be recorded:
   - **basic** records trace messages for security service set up and configuration events.
   - **intermediate** records the detailed security service run-time information in addition to the **basic** security service trace information. Note that recovery instructions are not displayed with each **intermediate** trace message.
7. To apply the trace level and close the Object Editor window, click the **OK** button.
8. For an application server, the recording of trace information starts immediately, without having to restart the application server.

For a client style, to start recording of trace information for individual clients you must restart those clients.

For examples of security trace log entries, see Security Trace Examples (page 57).

**Setting Security Trace for Long Term Tracing**

To set the security trace level for more long-term tracing of application servers or client styles, use the System Manager user interface to complete the following steps:

1. Display the System Manager user interface, and set the user-level to **Expert**.
2. Expand the **Management Zones** folder.
3. Expand the Management Zone for the application environment that contains the application servers or client styles.
4. Expand the **Configurations** folder.
5. Expand the Configuration that contains the application servers or client styles.
6. Set the security trace level attribute.

   To set the trace level for all members of a server group, complete the following steps:
   a. Expand the **Server Groups** folder.
   b. From the pop-up menu of the Server Group, click **Edit**.
      This displays the Object Editor for the Server Group.

   To set the trace level for a freestanding application server, complete the following steps:
   a. Expand the **Servers (freestanding)** folder.
   b. From the pop-up menu of the Server (freestanding), click **Edit**.
      This displays the Object Editor for the Server Image.

   To set the trace level for a client style, complete the following steps:
   a. Expand the **Client Styles** folder.
   b. From the pop-up menu of the Client Style, click **Edit**.
      This displays the Object Editor for the Client Style.
7. In the Object Editor window, click the **Object Services Trace** tab.

8. Set the **security trace level** attribute value to **basic** or **intermediate**, to enable the following trace information to be recorded:
   - **basic** records trace messages for security service set up and configuration events.
   - **intermediate** records the detailed security service run-time information in addition to the **basic** security service trace information. Note that recovery instructions are not displayed with each **intermediate** trace message.
9. To apply the trace level and close the Object Editor window, click the **OK** button.
10. To implement all trace level changes within a Configuration, activate the Configuration again.

    To do this, on the pop-up menu of the Configuration, click **Activate**.

    The System Manager displays an Action Console window that you can use to monitor the progress of the **Activate** action. The console first displays messages about the System Manager verifying that the Configuration is valid. If you have completed the above steps properly, you should see a *Configuration valid* message. The console then displays messages for activating parts of the Configuration. Finally, if you have completed the above steps properly, you should see a *Activation successful* message.
11. For an application server, the recording of trace information starts immediately, without having to restart the application server.

    For a client style, to start recording of trace information for individual clients you must restart those clients.

For examples of security trace log entries, see Security Trace Examples (page 57).

**Setting the Trace for All Application Servers**

To trace all application servers and client styles on a host, change the SecurityTraceLevel environment variable on that host; for example, by the following command:

```
set SecurityTraceLevel=2
```

Where the number indicates the trace level, as follows:

1. Security service set up and configuration events (basic trace)
2. Security service run time events (intermediate trace)

This enables trace information to be collected over a period of time *for all application servers and client styles on the host*. The trace level setting is preserved until the environment variable is reset, even if application servers or client styles are stopped and restarted or their system management Configuration activated again.

This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on the formatting of trace logs.

**Security Trace Examples**

The following examples outline security problems indicated by trace messages from the security service:

- Example 1: NO_PERMISSION Exception Raised by ORB in somoa_request_nothread (page 58)
- Example 2: Security Association Establishment Tracking (page 58)
- Example 3: Security DLL Loading Status (page 58)

**Example 1: NO_PERMISSION Exception Raised by ORB in somoa_request_nothread**
This exception is raised because no security context is received by the target server. A client sends a security message without a security context when security is disabled in the Client Image or the client fails to authenticate itself in the client security request interceptor.

The following messages are issued for this problem:

- With the security trace level set to at least **basic**, the following message is logged before the ORB's "NO_PERMISSION" message in the trace log file.

  ```
  SECURITY WARNING: No Security Context has been received
  ```

- The ORB logs NO_PERMISSION with the *hostname* of the client.
- In the activity log file of the client, one of the following messages is logged:
  - SECURITY TRACE: Security is disabled in System Management CDS data store.
  - SECURITY TRACE: Fail to authenticate security name = XXXX

To resolve this problem, make sure that the **security enabled** attribute (on the Security Service tab of the Object Editor notebook for the Client Style) is set to **yes** and that the client's user ID and password are correct.

**Example 2: Security Association Establishment Tracking**
Each client and server has to establish a security association between them when security is enabled. Most of the security problems happen in the process of establishing security associations.

With the security trace level set to at least **basic**, the following messages are logged:

- At the client side, the following two messages are logged:

  ```
  SECURITY TRACE: Invoke non_existent() on server server_name
  SECURITY TRACE: Complete non_existent() to server server_name
  ```

- At the server side, the following message is logged:

  ```
  SECURITY TRACE: Receive a new security service context from client client_name
  ```

**Example 3: Security DLL Loading Status**
If a conflict occurs between the security loading status and the security enablement status in a Client Style Image, and the security trace level is set to at least **basic**, the following messages are logged:

```
SECURITY TRACE: Security is enabled in System Management CDS data store.
SECURITY TRACE: Security Client DLL is loaded.
```

**Example 4: Multiple Threaded Security Establishment**
Security is designed to support a multiple threaded environment. However, only one thread is allowed to establish the security association between the client and server.

With the security trace level set to at least **basic**, the following message is logged when a security association is being established by another thread:

```
SECURITY TRACE: Session Entry with Target Server= server_name,
            Thread ID= ddddd
            is being established in the Client Vault Session Table.
            I(Thread ID= ddddd) am going to wait."
```

The following message will be logged when the security association has been established.

```
SECURITY TRACE:Wake up other threads that are waiting for the
           availability of session entry of the Target Server
           Target Server=server_name
```

**Example 5: Security Session Table Status Monitor**
The security associations between clients and servers are kept in the Security Session (Vault) Table. A session entry is searched in the Session Table before a new session is created in that table.

With the security trace level set to **intermediate**, the following message is logged when a session is searched in the Session Table.

```
SECURITY TRACE: Search Session Entry with
            Target Server= server_name,
            Server UUID= uuid_value,
            Thread ID= thread_id
            in the Client Vault Session Table.
```

**Example 6: Refresh the Server Credential**
The authn_and_refresh thread is spawned to refresh the server process DCE credential every 9 hours. To track the server DCE credential, with the security trace level set to at least **basic**, one of the following messages is logged when the refresh thread succeeds or fails to refresh server DCE credential.

```
SECURITY TRACE: Succeed to refresh server.
SECURITY TRACE: Fail to refresh server.
```

**Example 7: Multiple Hosts, with One Host in the Wrong DCE Cell**
In a multi-host environment where one of the hosts is configured (incorrectly) into a separate DCE cell, the message sent from this host to any other hosts in the Component Broker cell will generate the following error message:

```
gss_accept_sec_context, invalid DCE credential
```

**RELATED TASKS**
“Troubleshooting Security Problems” on page 74
“Appendix I. Security Messages” on page 279
Set Trace Levels (*System Administration Guide*)

# Chapter 14. APPC Communications Trace

Use this procedure to trace and display information about the flow of control for an application server using APPC to communicate with tier-3 systems. These trace logs are to be used by IBM Support team when you report problems related to APPC communications.

This topic describes how to turn on the APPC Communications trace and interpret the trace output:



### Setting APPC Communications Trace

To trace the APPC flow of control, set the trace level on the Server Image through the System Manager user interface. You have to set the PAA Communications trace level and the Transaction Service trace level for an application server. Follow these steps to turn on the trace:

1. Display the System Manager user interface, and set the user-level to **Expert**.
2. Expand the **Host Images** folder.
3. Right click on the Server Image for the application server that you are interested in, and select **Edit**. This displays the Object Editor for the Server Image.
4. In the Object Editor window, click the **Component Trace** tab.
5. Set the **PAA trace level** and **PAA communications trace level** attribute values to **advanced**.
6. Click the **Object Services Trace** tab.
7. Set the **transaction service trace level** attribute value to **Advanced**.
8. To apply the trace levels and close the Object Editor window, click the **OK** button.

This enables trace information to be collected over a period of time into files in the subdirectory service/server/*ServerName*. Refer to “showlog Utility” on page 68 for more information on the formatting of trace logs.

**Note:** If the system management Configuration that contains the model for the server is activated again, the trace level changes for the Server Image are reset.

### Environment Variables Used By Transaction Service

The Component Broker Transaction Service uses the following environment variables. Refer to the the *Component Broker Planning, Performance, and Installation Guide* for more information on default settings.

- **SOMCB_APPC_LIBRARY_NAME**
  Specifies the name of the IBM Communication Server APPC DLL which is used to send or receive requests over SNA.
- **SOMCB_NOF_LIBRARY_NAME**
  Specifies the name of the IBM Communication Server Node Operations Facility DLL which is used to monitor the status of the SNA node and create the SNA configuration.
- **SOMCB_APPC_PARTNER_LIVES**
  Specifies the number of times a server should be restarted before it deletes an

APPC Connection that is neither defined in the server's configuration nor required to complete outstanding transactions.

- **SOMCB_APPC_TIMEOUT**
  Specifies the maximum number of seconds an allocate or receive request is permitted to take.
- **SOMCB_APPC_DEFAULT_MODE_NAME**
  Specifies that the SNASVCMG mode group should be used for exchange log names (XLN) requests at server startup.

### Sample Trace Calls to IBM Communications Server

You should submit the formatted trace logs to IBM Support Team for analysis when you report problems related to APPC Communication. You would not have sufficient information to interpret the data in these logs. For example, the following trace shows calls issued by an application server to the IBM Communications Server.

```
ComponentId:    262253
ProcessId:      518
ThreadId:       422
FunctionName:   OTSAPPCLibraryNT::callAPPC(long,CORBA::Long)
ProbeId:        1487
SourceId:       1.6 src/objsvcs/transactions/ots_appc/otsalib_nt.cpp
Manufacturer:   IBM
Product:        Component Broker
Version:        2.0
SOMProcessType: 5
ServerName:     testsrv
clientHostName:
clientUserId:
TimeStamp:      10/30/98 12:54:11.100491287
UnitOfWork:     30229:boyle2
Severity:       3
Category:       3
FormatWarning: 0
PrimaryMessage: The function OTSAPPCLibraryNT::callAPPC(long,CORBA::Long):1487
  reported trace message.
ExtendedMessage:
RawDataLen:     164
RawData:
0000  07 00 00 01 00 00 00 00 - 00 00 00 00 04 00 CA C8  ................
0010  00 00 00 00 04 00 79 C8 - 00 00 00 00 00 00 00 00  ......y.........
0020  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0030  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0040  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0050  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0060  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0070  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0080  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
0090  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00A0  00 00 00 00                                        ....
```

The following trace is the response from IBM Communications Server to the calls in the preceding trace example. This shows you the return codes and any returned information. However, Component Broker does not trace the application data sent and received.

```
ComponentId:    262253
ProcessId:      518
ThreadId:       422
FunctionName:   OTSAPPCLibraryNT::callAPPC(long,CORBA::Long)
ProbeId:        1489
SourceId:       1.6 src/objsvcs/transactions/ots_appc/otsalib_nt.cpp
Manufacturer:   IBM
Product:        Component Broker
Version:        2.0
```

```
SOMProcessType: 5
ServerName:     testsrv
clientHostName:
clientUserId:
TimeStamp:      10/30/98 12:54:11.101889230
UnitOfWork:     30229:boyle2
Severity:       3
Category:       3
FormatWarning: 0
PrimaryMessage: The function OTSAPPCLibraryNT::callAPPC(long,CORBA::Long):1489
   reported trace message.
ExtendedMessage:
RawDataLen:     164
RawData:
0000  07 00 00 01 00 00 00 00 - 00 00 00 00 04 00 CA C8  ................
0010  00 00 00 00 04 00 79 C8 - 00 02 D3 E4 F6 F2 D7 E2  ......y.........
0020  40 40 C7 C2 C9 C2 D4 C9 - E8 C1 C9 E8 C1 F7 E3 F2  @@..............
0030  F9 F0 49 59 41 37 54 32 - 39 30 49 59 4F 4D 34 31  ..IYA7T290IYOM41
0040  30 20 C9 E8 D6 D4 F4 F1 - F0 40 00 00 C7 C2 C9 C2  0 .......@......
0050  D4 C9 E8 C1 4B C9 E8 D6 - D4 F4 F1 F0 40 00 40 40  ....K.......@.@@
0060  40 40 40 40 40 40 40 40 - 01 00 D7 C7 08 04 00 00  @@@@@@@@........
0070  C8 94 45 97 E1 00 00 00 - 00 00 00 00 00 00 00 00  ..E.............
0080  00 00 11 C7 C2 C9 C2 D4 - C9 E8 C1 4B C9 E8 C1 F7  ...........K....
0090  E3 F2 F9 F0 DF E9 AD 35 - 03 00 00 01 00 0B 84 57  .......5.......W
00A0  7D CF 80 83                                       }...
```

**RELATED TASKS**


“Troubleshooting APPC Problems” on page 77
“Example: APPC Messages in the Activity Log” on page 81
“Appendix H. APPC Messages” on page 263
“Investigating Communications Server Problems” on page 88

# Chapter 15. Problem Determination Tools

Several tools are available to help you with Component Broker problem determination. They are:

- "Sherlock Tool"
  Processes a formatted activity log against a database of known problems to obtain information on workarounds and fixes.
- "showlog Utility" on page 68
  Formats the activity log, error log, event log, and some of the component and object services trace logs into readable text.
- "bgtrfmt Utility" on page 69
  Formats the agent trace log files for viewing.
- "Model Consistency Checker" on page 70
  Verifies models that you have built with the Object Builder.

## Sherlock Tool

The Sherlock tool processes an activity.log file against a database of known problems. It generates an output file containing an analysis of errors and exceptions found in the activity log. The analysis provides suggested user response and workaround information. Every time the Sherlock tool finds a match for an error in the activity log in its database, it writes the database's information into the analysis output file.

Contact your IBM Engagement Team representative or IBM Support for more information on the Sherlock tool.

**RELATED REFERENCES**
"Sample Input Activity Log to Sherlock"
"Sherlock Output Sample" on page 67
"Chapter 2. Activity Log for Problem Determination" on page 3
"Chapter 16. Troubleshooting Component Broker Run-Time Problems" on page 71

## Sample Input Activity Log to Sherlock

This is a sample of the formatted activity log that is processed by the Sherlock tool. Your activity.log file must be formatted with the **showlog** utility prior to submitting it to IBM. You will need to refer to this log when you review the "Sherlock Output Sample" on page 67.

```
------------------------------------
ComponentId: 103
ProcessId: 397
ThreadId: 545
FunctionName: SOMSR_ContinueDispatchCallbackObject::wrapExecute(void*)
ProbeId: 530
SourceId: 1.20.1.3 src/sr/somsrcb.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
```

```
TimeStamp: 9/17/98 17:29:30.072302547
UnitOfWork: 398:kewegner
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
   SOMSR_ContinueDispatchCallbackObject::wrapExecute(void*):530 reported an
   activity.
ExtendedMessage: Thread Manager caught Unknown Exception from continue_dispatch
RawDataLen: 0
-----------------------------------
ComponentId: 103
ProcessId: 397
ThreadId: 545
FunctionName: SOMSR_ContinueDispatchCallbackObject::dumpTargetInfo(void*)
ProbeId: 301
SourceId: 1.20.1.3 src/sr/somsrcb.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/17/98 17:29:30.077448451
UnitOfWork: 398:kewegner
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
   SOMSR_ContinueDispatchCallbackObject::dumpTargetInfo(void*):301
   reported an activity.
ExtendedMessage: SRThreadManagerUnknownError: additional information ->
   —>Method Name:  addBeneficiary
   —>Target classname:  PolicyEmSQLMO
   —>Target type_id:  IDL:PolicyEmSQLMO:1.0
   —>Target refcount:  1
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 461
ThreadId: 56
FunctionName:
IBOIMLocalToServer_IMMixinBase::setCachingManagedObjectFlag(CORBA::Boolean)
ProbeId: 3428
SourceId: 1.105 src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/17/98 16:44:15.824745715
UnitOfWork: 28381:kewegner
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
   IBOIMLocalToServer_IMMixinBase::setCachingManagedObjectFlag(CORBA::Boolean):3428
   reported an activity.
ExtendedMessage: A managed object (class name: UNKNOWN) does not
   match the caching policy of the container. The reason code is : 0.
   Explanation of the reason codes : 1 - The  container has the
   dataCachedInManagedObject set to YES, but the managed object does
   not inherit from the IManagedServer::IManagedObjectWithCachedDataObject.
   2 - The container has the dataCachedInManagedObject set to NO, but the
```

```
   managedObject does not inherit from the
   IManagedServer::IManagedObjectWithDataObject. 3 - The managed object did
   not support  either interface, IManagedServer::IManagedObjectWithCachedDataObject
   or  IManagedServer::IManagedObjectWithDataObject
RawDataLen: 0
-------------------------------------
ComponentId: 262251
ProcessId: 607
ThreadId: 472
FunctionName: osql::osql_exec_lazy(char*)
ProbeId: 1172
SourceId: 1.20 src/objsvcs/query/impl/osql.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/17/98 16:23:48.315355880
UnitOfWork: 497:kewegner
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function osql::osql_exec_lazy(char*):1172 reported an error.
ExtendedMessage: Query = "select x from collection1 x where poxlicyNo>3;;"
   Query error message = "Error - Unresolved column poxlicyNo has been found"
RawDataLen: 0
-------------------------------------
.
.
.
```

**RELATED REFERENCES**

"Sherlock Tool" on page 65
"Sample Input Activity Log to Sherlock" on page 65

## Sherlock Output Sample

The following is the contents of a sample output file from Sherlock. Use an editor (that shows line numbers) to browse the formatted input activity log and match each of the Sherlock comments to the related line number in the activity log. See "Sample Input Activity Log to Sherlock" on page 65 for the sample activity log for the following Sherlock output.

```
-----------------------------------------
*** Symptom tmunex.txt found in record 1 (line 1)
This message is a signal that a method raised an exception which
was not part of its raises clause.  Remember that only system exceptions
and the exceptions which are part of the raises clause can be returned by
a method.  There should be a dumpTargetInfo message shortly after this
message.  It will contain the information pertaining to the method
which raised the exception.  If that method is customer code, it would
be useful to to instrument that method by putting in additional
try - catch blocks to determine what is throwing the exception.
If there is a handleSignal message immediately preceding this
message, it may be that the method has caused a segment violation.
If the handleSignal message provides a call stack, this may give
you additional clues on how to diagnose the problem.
*** Symptom  unknown found in record 1 (line 1)
Sherlock cannot identify this problem.
*** Symptom  unknown found in record 2 (line 23)
Sherlock cannot identify this problem.
*** Symptom mocntr.txt found in record 3 (line 49)
```

```
This is a warning message.  The mixin code automatically fixes the
problem.  However, the user should be aware that one of three conditions
exists:
1 - The  container has the dataCachedInManagedObject set to YES, but the
    managed object does not inherit from the
    IManagedServer::IManagedObjectWithCachedDataObject.
2 - The container has the dataCachedInManagedObject set to NO, but the
    managedObject does not inherit from the
    IManagedServer::IManagedObjectWithDataObject.
3 - The managed object did not support  either interface,
    IManagedServer::IManagedObjectWithCachedDataObject or
    IManagedServer::IManagedObjectWithDataObject
This problem should be fixed by adjusting the container that the object is
put in to one which will support the correct caching strategy for the MO.
.
.
.
```

**RELATED REFERENCES**

"Sherlock Tool" on page 65

## showlog Utility

The showlog utility is used to format the activity log, error log, event log, and the component and object services trace logs into readable text. Run the showlog tool on the host that had the error to get the optimum substitution values in the output file. You can also browse the activity and error logs from the System Manager user interface without using the showlog utility.

### showlog Command Syntax

The **showlog** command has the following syntax:

```
showlog -nt | filename [-debug]
```

WIN The **-nt** option formats the error log (only valid on the Windows NT platform).

*filename* is the input log file, for example, activity.log. The showlog utility reads the file and formats it for reading.

**-debug** formats the entry with full debug information (this option is very useful when debugging a system problem). **Note**: You must specify **-debug** *after* the file name; otherwise, you will get a formatted debug version of the event log.

If **-debug** is not specified, only **PrimaryMessage**, **ExtendedMessage**, and **RawDataLen** information is included for each entry in the log; for example, you get the following information:

```
{Activity Warning}
The function IBOIMManagedObjectCustomization_ICDSDataObjectImpl::getStringByName
(const char*):940 raised CORBA exception
IBOIMManagedObjectCustomization::ICDSDataObject::IAttributeNotFound, error code is
0x0 0.
An exception occurred while accessing CDS image
HostIkimberlySomServIkimberly Name
ServerSomContainerIiCDSCachedHomes, attribute usesPolicyGroup. Reason code 1.
Reason codes: 1 - The attribute usesPolicyGroup not found. 2 - The cursor was not
open for the image. 3 - A request was not made to get the data for attribute,
usesPolicyGroup as the wrong type. 4 - CDS exception id: 110028, exception text:
No Attribute of that name was found. 5 - An unknown exception.
```

See the “Chapter 2. Activity Log for Problem Determination” on page 3 for more information on all the fields of each entry in the log.

**Note**: If the output is piped to a file and an editor is used to display the information, turn on word wrap in the editor to view the full contents of a line.

**showlog Example 1**

```
showlog c:\CBroker\service\activity.log -debug > showlog.out
```

Produces the text file named showlog.out from the activity log with all information included as part of each entry.

**Showlog Example 2**

```
showlog -nt
```

Produces a text file from the event log; these are abbreviated entries (not all the debug information is included for each entry).

**Formatting Multiple Trace Files**

When you set one or more types of component or object service trace, multiple trace files are generated. The trace files are in the service/server/*ServerName* subdirectory of the host on which the application server runs. The names of the trace files have a *yydddhhmmss.xxx* format where:

- *yy* is the year
- *ddd* is the Julian date
- *hh* is the hour
- *mm* is the minutes
- *ss* is the seconds
- *xxx* is a three digit number that runs from 101 to 999 and then rolls over to 101

98279095414.111 is an example of a trace file name. A single trace file is created for each ORB request. You may chose to write a script file to sequentially run **showlog** against each of the trace files (ascending sequence) and concatenate them (using >>) into a single file. If you have set more than one type of component or object service trace, all trace information is merged into the same set of files.

**RELATED REFERENCES**

“Chapter 15. Problem Determination Tools” on page 65

## bgtrfmt Utility

To format the agent trace log files (.dct, .tr1, .tr2 files) in the ../data directory, use the **bgtrfmt** utility. The syntax for the utility is:

```
bgtrfmt filenamestem > outputfilename
```

where *filenamestem* is the file name without the file extension and *outputfilename* is the output file for the formatted trace information. If you have trace files, say, bgmain_mySystem.dct, bgmain_mySystem.tr1, and bgmain_mySystem.tr2, you would invoke:

```
bgtrfmt bgmain_mySystem > bgtrace.output
```

That is, supply the stem of the file names without the .dct, .tr1, or .tr2 extension, and redirect the output to some file so you can look at it.

WIN When running on NT, you will not be able to run **bgtrfmt** if the trace files are still open by the System Management executable that created them. You will have to either stop the executable or create a temporary directory and copy the three trace files to that directory. You may find the latter to be simpler.

**RELATED REFERENCES**
“Chapter 5. Agent Trace Log for Problem Determination” on page 19
“Chapter 15. Problem Determination Tools” on page 65

## Model Consistency Checker

You can use the Model Consistency Checker tool to verify models that you have built with the Object Builder. The Model Consistency Checker provides a mechanism for you to identify problems, such as dangling relationships, in your model before compiling or running your application. You can run this tool either from the command line or from the Object Builder’s GUI. See related tasks for more information on how to run the tool.

**Note:** If you are collecting data to report a problem to IBM, turn on all the optional checking when running the Model Consistency Checker.

**RELATED TASKS**
Check a Model for Consistency (*Application Development Tools*)
“Chapter 15. Problem Determination Tools” on page 65

**RELATED REFERENCES**
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72

# Chapter 16. Troubleshooting Component Broker Run-Time Problems

In this section, recommendations are provided to help you troubleshoot some of your Component Broker run-time problems.

**Read the Activity Log**
Always look at the entries in activity log when you are diagnosing run-time problems. Follow these steps:

1. Read the PrimaryMessage and ExtendedMessage information to help you understand the error condition. If you find minor code messages (0x4942*xxxx*) in the PrimaryMessage field, look them up in the “Appendix G. System Exceptions and Minor Codes” on page 217 to get a detailed explanation of the error and the suggested user response.
2. Trace the resulting error listed in the activity log to the initial exception that caused the error. This may help you identify the reason for the initial failure. See “Reading the Activity Log” on page 6 for more information on how to do this.

When you have completed the above steps and you still do not know what could have caused the problem, you can review the following problem scenarios and try some of the suggested actions to help you diagnose the error.

- “Troubleshooting Component Broker Service Start Up”
- “Troubleshooting Server Activation Problems”
- “Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
- “Troubleshooting Cache Related Problems” on page 74
- “Troubleshooting APPC Problems” on page 77
- “Troubleshooting Security Problems” on page 74

**RELATED REFERENCES**
“Chapter 1. Problem Determination Information” on page 1
“Chapter 17. Problem Determination Hints and Tips” on page 103

## Troubleshooting Component Broker Service Start Up

If you cannot start up the Component Broker Connector Service from the Service dialog in Control panel, the password registered for the service may be wrong. Did you change your password recently? If so, try changing the password in the Services account dialog.

**RELATED REFERENCES**
“Chapter 1. Problem Determination Information” on page 1
“Chapter 16. Troubleshooting Component Broker Run-Time Problems”

## Troubleshooting Server Activation Problems

When you get a failure while starting up a server, try one or more of the following actions:

- Check to see if the Name Server started up correctly in the activity log. Start reading the log from the point of activation all the way through to the failure.
- Look in the DCE director for any sign of the server.
- If the activation of a configuration failed without an error message, for example, verification is complete but the server activation did not start, check to see if the servers are up (in DCE). Does your application use security? If yes, disable security on the agent client style image (security on agent client image is automatically turned on), bring down the Component Broker Connector service and bring it back up to let it take effect. Refer to “Troubleshooting Security Problems” on page 74 for more information on diagnosing security problems.
- Check if the ORB daemon has been started.
- Start with a clean activity log. If you have been starting up a few servers with a single activation, try starting them up one at a time.
- Follow the instructions in Hints and Tips: Activity Log (page 103) to format and erase the activity log so that you start with a clean log. If the server that has the problem is still running, disable the server by stopping it. Define a new server and reactivate it. Configure the application on the new server and see if the problem is repeated.
- Is the system very slow and the server activation always failing? Are there also timeout problems with CORBA exceptions and errors? Consider increasing the paging file size ( WIN for example, to 400 or 500MB). Verify if the DNS IP address is incorrectly set. You may also want to change some of the server timeout settings. Refer to server timeout settings (page 106) for more information on these settings.
- If you get segmentation faults when performing an activation in System Management and you have Java server applications, check the PATH system environment variable. The JDK directory (for example, JDK1.1.6) should be in the front of the PATH system environment variable to avoid conflicts with other products that may contain Java DLLs (such as Lotus products). See the JDK Path Setup (page 108) for more instructions on how to do this.

**RELATED REFERENCES**

*Planning, Performance, and Installation Guide*

“Chapter 1. Problem Determination Information” on page 1
“Chapter 16. Troubleshooting Component Broker Run-Time Problems” on page 71

## Troubleshooting Unexpected Application Error or Bad Data Problems

When the client application gets an unexpected error or bad data, try one or more of the following actions:

- When you get a CORBA:UNKNOWN error and, if in the activity log, you see an entry where a database connection cannot be made, check to see if the database (for example, DB2) has been started. Refer to “Activity Log Sample 4: Client Application Receives an Error” on page 10.
- If your client application fails with this message:

```
makeChildren Exceptionorg.omg.CORBA.UNKNOWN: minor
   code: 1229062304 completed: Maybe
```

  And the activity log shows this DB2 failure:

```
PrimaryMessage: The function db2emb_access::check_sqlca():524
raised CORBA exception IBOIMException::IDataObjectFailed,
 error code is 0xFFFFFC46 rc.
  ExtendedMessage:
*** SQL error code: -954, SQLSTATE: 57011, (...)
```

Then, possibly not enough storage is available in the application heap. Refer to Run-Time Environment Settings (page 106) for more information on changing the DB2 heap size.

- If you are getting a CORBA::PERSIST_STORE error, it is very likely that the problem is related to SQL. The ExtendedMessage in the activity log entry for the initial error often contains SQL information that is directly related to the cause of the failure. Look up the SQL error in the SQL message guide (in the DB2 online documentation). You can probably debug most SQL errors just by using this guide. If you still require more information on the SQL statement, run Query Service Trace.
- If the client application hangs (if timeout is set to infinite) or ends with a CORBA::NOTRANSACTION exception, and the application server is not running, see if the transaction did not start (look in the activity log). If the transaction did not start, check your code to see if a begin transaction was executed prior to the query. If the begin transaction was executed, put a try catch around it to see if it completed successfully. See the *Advanced Programming Guide* for more information on Application Programming Using Transaction Service.
- If you want to have a quick look at the application flow, turn on the ORB request trace, clean the activity log, and rerun the application.
- If you suspect that the problem is in the user code, run OLT.
- If you suspect that the problem is not in the user code, you can run the XYZ trace, the ORB communication trace or both of them together.
- If the same model is generating different run-time errors, make sure that you run the Model Consistency Checker to verify your model.
- You can submit your formatted activity log to run against the Sherlock tool to see if there are workarounds for your problem.

**RELATED REFERENCES**

“Chapter 8. Query Service Trace” on page 27
“Chapter 9. Cache Service Trace” on page 33
“Chapter 6. Object Level Trace for Problem Determination” on page 23
“Chapter 10. ORB Request Trace” on page 39
“Chapter 11. ORB Communication Trace” on page 41
“Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace” on page 51
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72
“Troubleshooting Cache Related Problems” on page 74
“Chapter 2. Activity Log for Problem Determination” on page 3
“Sherlock Tool” on page 65
“Model Consistency Checker” on page 70
*Planning, Performance, and Installation Guide*

**RELATED TASKS**

Configure DB2 to use the CBConnector Data Cache (*System Administration Guide*)
Configure a Cache for SQL Data (*System Administration Guide*)

## Troubleshooting Cache Related Problems

If your activity log contains a system exception with a minor code (somewhere within the range of 00FA-012B), then you have an error related to the data cache service. You can try one or more of the following:

- If the major code is BAD_PARAM, INV_ORDER or BAD_TYPECODE, then the code generated for the DataObject Implementation that calls the cache service has made a bad call or has passed a bad argument to the cache service. One possible cause of this error is that the relational schema information contained in the DDL is inconsistent or has become inconsistent with the actual database.
- If the exception is related to an SQL error, the cache service logs the failing SQL statement along with the failing sqlcode in the activity log. Examine the activity log to see if there is such a SQL error just prior to the exception.
- Make sure you have bound the two .bnd files to the target database (see "Configure DB2 to Use the CBConnector Data Cache" in the *System Administration Guide*) and that the userid that the CBConnector server is using has the authority to use these packages and access your database and read and update your application tables.
- Make sure that your application has issued a begin() transaction before attempting to access any objects that are stored in a DB2 database.

If you require additional information to help you diagnose the problem, you can use the cache service trace.

**RELATED CONCEPTS**
Data Cache Considerations (*System Administration Guide*)

**RELATED REFERENCES**
“Appendix G. System Exceptions and Minor Codes” on page 217
“Chapter 8. Query Service Trace” on page 27
“Chapter 9. Cache Service Trace” on page 33
“Troubleshooting Unexpected Application Error or Bad Data Problems” on page 72

**RELATED TASKS**
Configure DB2 to use the CBConnector Data Cache (*System Administration Guide*)
Configure a Cache for SQL data (*System Administration Guide*)

## Troubleshooting Security Problems

If you encounter a security problem when running applications with security enabled, you can use the following checklist to help isolate and resolve the problem.

1. Check the Component Broker activity and error logs for security messages.
2. Check for common security problems (page 74).
3. Check the DCE configuration (page 75).
4. Check security and run-time setup (page 76).

**Common Security Problems**
Some common problems that you may encounter when setting and administering the Component Broker security system are:

- Server Authentication (page 75)

**Server Authentication**
Server is authenticated with the DCE Security server. A server normally authenticates itself without operator intervention. A transient failure in communication may prevent the server authentication, and errors will occur when the clients invoke methods on objects in the server.

Also, login or certificate information that authenticates the server is stored in files on the server. If the file system is corrupted, problems will occur when the server is unable to access the required information for authentication.

Configuring the keytab file is automatically done by the Component Broker when the server is created. However, if the keytab file is corrupted, you must manually recreate the keytab file using the DCE administration tools. Note that DCE replication service makes a backup copy of the keytab file. Also, you must periodically update the server's password and synchronize the passwords between the user registry and the keytab file. You can use the DCE's facilities to automatically generate passwords. This is a more secure procedure because the passwords will only be known to the DCE and Component Broker servers.

Server keyrings are also a potential source of problem. If, for some reason (for example password expiration), you have to recreate or renew the certificate, depending on how this affects the server's trust basis, you may have to redeploy the client keyrings to cite the new trust-basis.

To help prevent any compromise, be sure to set the permissions on the server's keyring so that only the server and the relevant administrators can access it.

**Configuration Attributes**
Component Broker has a large number of configuration attributes to provide fine-grained control over the behavior of security by editing client images and server images. However, these choices also provide an opportunity to incorrectly set up security. Most of the invalid choices are caught and flagged during Component Broker system management configuration. However, some of the more advanced configuration options, such as the qualities of protection (QOP) models, are not evaluated until run time and related errors will not be caught until then.

**Access to Client Keyring Class**
The value that you specify in the **keyring file** attribute of the Client Style is a Java class name. Java and the Component Broker run time expect to find this keyring as a Java class in the classpath, codepath, archive, or applet URL of the Java client application or applet. If you placed your client keyring in the Component Broker keyrings directory, make sure that the class you specified for the keyring in IKMGUI is the same as the name you specified in the **keyring file** attribute on the Client Style. These names must have identical spelling (with same package prefix), and they are case-sensitive.

If you are working with a Java client applet, make sure that the keyring is accessible from your Web Server, and that the codepath, archive, or applet URL include the location of your keyring class. Also copy the client-style properties file and client keyrings to the Web Server host. Make sure that your client applet correctly refers to the URL in this properties file in a com.ibm.CORBA.ClientStyleImageURL parameter of the HTML applet tag.

### Check DCE Configuration

Because Component Broker supports only one DCE cell, ensure that all DCE clients and their DCE servers are configured into one cell. Component Broker Server is a DCE client so you can have more than one Component Broker Server per DCE cell.

- WIN For Windows NT, complete the following steps:
  1. From the Windows NT **Start** menu, click **Programs - DCE for Windows NT - DCEsetup**.
  2. In the right panel of DCE Setup, ensure that the **Cell Name** entry is consistent with the same entry on the other hosts within the cell.
  3. For a configured DCE server, ensure that the following services are **enabled** and in **running** state:
     - Security Client
     - Security Server
     - CDS Advertiser
     - CDS Server
     - DTS
  4. For a configured DCE client, ensure that the following services are **enabled** and in **running** state:
     - Security Client
     - CDS Advertiser
     - DTS
- AIX For AIX, complete the following steps:
  1. From a shell prompt, enter:

     ```
     lsdce
     ```

     to list all configured DCE services. Each service should be in **Complete** state.
  2. Enter

     ```
     dce_login
     ```

     to ensure that the DCE services are running.
  3. Enter

     ```
     getcellname
     ```

     to return the DCE cell name for each host. Ensure that all hosts have the same cell name.

### Check Security and Run Time Setup

Follow these steps to verify that the necessary setup for security service has been completed:

1. Check the security enablement.

   From the System Manager user interface, right click on each entry listed under Client Style Images and Server Images and select **Edit**. In the Object Editor, select the **Security Service** tab, and ensure that the **security enabled** field is set to **yes**.
2. Check that you have created appropriate certificates for servers and added them into the keyrings for your client styles.
3. Check that you have distributed the server and client keyrings onto appropriate hosts in your enterprise.

4. Check that you have made the client properties files available to the clients that need them.
5. Check that you have set up appropriate accounts for your client and server principals.
6. If you still suspect a security-related problem, but need more information, you can turn on Component Broker trace for the security service on one or more application servers. This enables security trace information to be collected over a period of time. To turn on security service tracing, see “Chapter 13. Security Service Trace” on page 55.

**RELATED TASKS**

“Appendix I. Security Messages” on page 279
Qualities of Protection (*System Administration Guide*)
Create and Install Server Certificates (*System Administration Guide*)
Place Server and Client Keyrings in your Enterprise (*System Administration Guide*)
Administer Accounts for Client and Server Principals (*System Administration Guide*)
Enable Security within a Configuration (*System Administration Guide*)
Create and Protect Server Keytab Files (*System Administration Guide*)

## Troubleshooting APPC Problems

This topic provides information about diagnosing problems in the Component Broker Transaction Service when it is communicating with a tier-3 system over APPC.

Tier-3 communication over APPC involves a number of products, each with its own terminology and producing its own diagnostics. Therefore a large part of the problem determination effort is piecing together information from different sources. This task is easier if you understand the function of each product and are familiar with its diagnostics and symptoms of common problems.

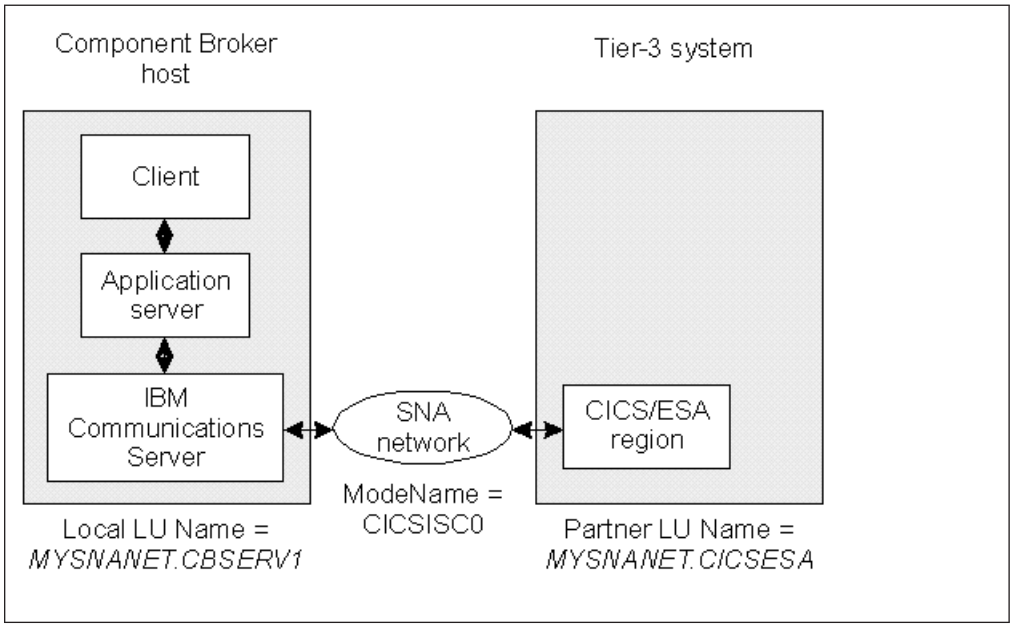
The aim of this topic is to help you with the APPC problem determination process by explaining the available diagnostics information and its interpretation.

The general sequence of tasks when investigating APPC problems is as follows:

1. Identify the products or components (page 77)
2. Determine the symptoms (page 78)
3. Identify the failing component (page 79)
4. Trace the failing request (page 80)
5. Check the configuration of components (page 80)

**Identify the Products and Components**

As a first step, draw out the components being used. For example, the diagram below shows a typical PAA APPC set up where a client is communicating via a Component Broker application server to a CICS/ESA region. Alternatively, the tier-3 system may be a Transaction Series server or an IMS region.




Drawing this sort of diagram is often a good place to start, especially if you are diagnosing a problem on a system that you did not set up. It is worth spending a little time identifying the systems and products and their configuration information (for example, system names, mode names, and so on).

For more information about determining the systems and products used for APPC communication in your Component Broker network, see "Collect Information for Your SNA Configuration" in the *Systems Administration Guide*.

**Determine the Symptoms**

Next, try to identify the symptoms of the failure. For example, try to identify symptoms related to the following points:

- What was the application server or tier-3 system doing when the failure occurred? Was it initializing or had it been running for some time?
- What applications were running at the time of the failure? What should they do?
- When does it fail, all the time or intermittently? Has it ever worked and, if so, what were the circumstances?
- How does the application fail? Does it hang? Is it looping? Was an unexpected exception raised? Or, was there a server crash?
- Did the application start to fail after a particular event such as a change to the configuration or a system crash (such changes are often the cause of the problem)?
- What are the steps required to recreate the problem?

If possible, try to do a little experimentation to narrow down the failure to a simple scenario, or to discover the extent of the problem. Try to understand if all PAA APPC requests fail, or just those sent to a particular system, or if the problem is confined to a single application or Component Broker server.

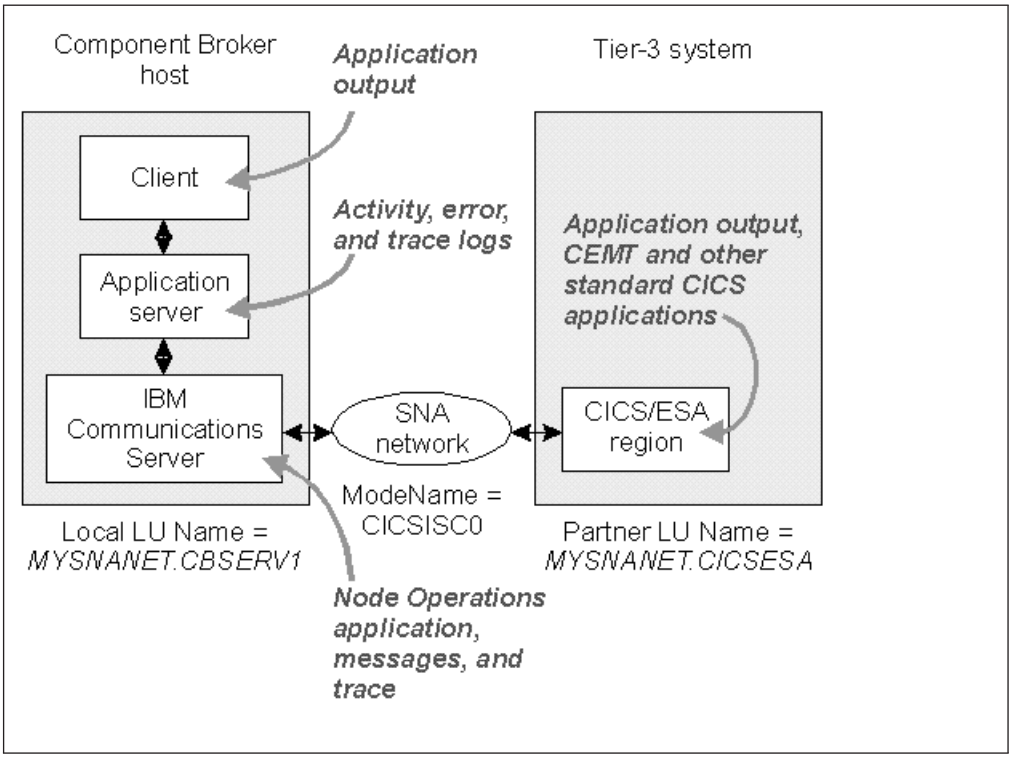
This type of analysis often indicates the type of problem you are dealing with, for example:

- If all PAA requests fail then it is likely that there is a configuration problem or that a component or product is not running.

- If requests suddenly stop working, then perhaps a change to the system (usually by operator intervention) may have caused the problem. Such changes may include configuration updates, deletion or relocation of system files, or relocation of systems to a different machine.
- Hangs tend to be caused by either insufficient resources in the network which includes the local Component Broker server and the tier-3 system, or deadlocks in the server or application.

**Identify the Failing Component**

Next, try to identify the failing component or product. For each component or product in your diagram, verify that it is running. If it is, check the message logs to see if one or more components have generated error or warning messages, especially during its startup and when the failing request occurred. The diagram below summarizes where to look for these messages.




Probably the easiest place to start looking for messages is the Component Broker application server's activity log, because it contains error messages describing, in Component Broker terms, any task it could not perform. Check the descriptions of any APPC error messages that you find.

However, it is possible that several components have reported errors. Usually the component that initially detects the error gives the most precise diagnostics on the cause of the problem. Subsequent messages probably describe what was affected by the initial failure. If the Component Broker messages just report a network error or a remote program failure, then you will need to look for messages in other products or components.

The following table provides a summary of the diagnostic information that is useful for different types of errors.

| Problem -> Information<br>\|<br>v | Appl Server Unable to Initialize | Appl Server Unable to Connect to Tier-3 System | APPC request fails | Hang during APPC request | Appl Server crashes | Handle Signal in Appl Server Activity Log |
|---|---|---|---|---|---|---|
| **Description of the problem** | Yes | Yes | Yes | Yes | Yes | Yes |
| **Activity log** | Yes | Yes | Yes | Yes | Yes | Yes |
| **Component Broker trace** | Yes | No | Yes | Yes | Yes | Yes |
| **IBM Comm Server* configuration file** | Yes | Yes | Yes | Yes | No | No |
| **IBM Comm Server* message file** | No | Yes | No | No | No | No |
| **IBM Comm Server* trace** | Yes | Yes | Yes | Yes | No | No |
| **Messages and diagnostics from tier-3 system** | No | No | Yes | Yes | No | No |
| **Transaction Service log** | Yes | No | No | Yes | Yes | No |

*IBM Communications Server

**Trace the Failing Request**

If the messages do not identify the problem, try turning on trace in each component or product and re-running the failing request. Traces show the calls and responses being made throughout the system and may display a bad parameter or result. Traces are often designed for developers of the component or product and are not easy to read without internal design and code information. However, traces can tell you if the request reached a particular component or product. If you find that the request failed before reaching its final destination, then concentrate your suspicions around the component that rejected the request. Recheck the configuration and also refer to the documentation on common errors for the component.

The following describes how you can turn on some of these traces:

- “Chapter 14. APPC Communications Trace” on page 61
- “Tracing Calls to the Communications Server” on page 89

**Check the State and Configuration of Components**

A common source of problems in APPC communications is the configuration of the components or products. If you have identified a failing component, check that it is running properly and that its configuration is correct. You should also check other components’ configurations to make sure that they are running properly and that you have configured them correctly because they may have a configuration problem that is impacting the failing component.

- Check the Status and Configuration of the SNA Network. (See “Checking the Status and Configuration of the SNA Network” on page 91.)
- Check the states of resources in a CICS region. (See “Checking the States of Resources in a CICS Region” on page 97.)
- Check that you have configured the APPC Connection to a tier-3 System correctly within Component Broker, as described in the *System Administration Guide*.
- Check that Communications Server names Match CICS definitions, as described in the *System Administration Guide*.

For more information about checking the state and configuration of tier-3 systems, see the information provided with those products.

**RELATED REFERENCES**
“Appendix H. APPC Messages” on page 263

## Example: APPC Messages in the Activity Log

Here are some examples of the messages that are generated by the PAA APPC support and written to the Component Broker’s activity log. If you experience problems with PAA APPC, you should browse through the application server’s activity log messages beginning with the entry on the server startup. Search for the string **ots_appc**, because the source of most PAA APPC messages are in files under this directory.

The following are examples of messages for:

- Successful startup of a server (page 81)
- An unexpected Return code from IBM Communication Server (page 86)
- Loading the Java Virtual Machine (page 87)
- SECURITY ERROR message on each PAA APPC request (page 87)

To look up the meaning of a particular transaction service APPC message, see “Appendix H. APPC Messages” on page 263.

**Successful Server Startup**
This example shows messages from an application server that successfully initializes PAA APPC. These messages are located in the formatted activity log. Search for the string, **ots_appc**.

1. The first message you would see is the APPC support being loaded into the server’s transaction service. This occurs during the running of the server’s initializers and is the result of Component Broker library somtra1i.dll being initialized.

```
_____________________________-
ComponentId: 262253
ProcessId: 505
ThreadId: 520
FunctionName: OTSAPPCSRM::OTSAPPCSRM()
ProbeId: 106
SourceId: 1.5 src/objsvcs/transactions/ots_appc/otsasrm.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
```

```
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:31.666955881
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function OTSAPPCSRM::OTSAPPCSRM():106 reported an activity.
ExtendedMessage: APPC support has been enabled in the transaction service
RawDataLen: 0
-------------------------------
```

2. Next, you would see the IBM Communication Server libraries being loaded. There are two libraries involved, wappc32.dll and winnof32.dll. They are loaded when they are first required. If you are running PAA APPC, this occurs when the first PAA APPC request runs. If the server has previously run PAA APPC requests, then the libraries are loaded during the initialization of the transaction service as the server starts up.

   Both of these libraries are loaded together. However, only the loading of the wappc32.dll is logged. Verify that the Component Broker is using the IBM Communications Server version of wappc32.dll rather than that of the Microsoft SNA Server. Check that the string, **IBM SNA**, appears in this message rather than **WinAPPC - SNA Server for Windows NT**. Microsoft SNA Server's wappc32.dll is *not* supported by Component Broker. Both servers use a DLL with the same name and only one can be installed on the host machine.

```
-------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 535
FunctionName: OTSAPPCLibraryNT::OTSAPPCLibraryNT()
ProbeId: 194
SourceId: 1.6 src/objsvcs/transactions/ots_appc/otsalib_nt.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:40.508869083
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCLibraryNT::OTSAPPCLibraryNT():194 reported an activity.
ExtendedMessage: The SNA library wappc32.dll, version 1.0, for IBM SNA has
  been loaded in the server.
RawDataLen: 0
-------------------------------
```

3. When a new local LU name is passed to the transaction service (when it is recovered from the transaction service log, or when a PAA APPC application makes use of an APPC Connection for the first time), the transaction service verifies that the local LU name contains valid values and is correctly defined to the IBM Communications Server. The following message indicates that the local LU Name is correct.

```
-------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 535
FunctionName: OTSAPPCListenerNT::checkName(OTSAPPCLUName*)
ProbeId: 255
SourceId: 1.7 src/objsvcs/transactions/ots_appc/otsalist_nt.cpp
```

```
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:40.854713277
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCListenerNT::checkName(OTSAPPCLUName*):255 reported an activity.
ExtendedMessage: Server CashAcctEMSvr will send APPC requests from SNA local LU name
'USIBMZP.PVT01001' which is defined with an alias name of 'PVT01001'
RawDataLen: 0
-------------------------------
```

4. The following activity log entry shows an APPC Connection being recovered from the transaction log. The transaction service automatically writes information about each APPC Connection it is using to the transaction log. This is required in case an APPC Connection is deleted from the server's configuration while the server is stopped with incomplete transactions. When the server is restarted, it uses the APPC Connection's information that was recovered from the transaction log to contact the partner SNA system and complete the transaction.

```
-------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 535
FunctionName: OTSAPPCPartner::recreateObject(OTSSRMLogSection*,CORBA::ULong)
ProbeId: 546
SourceId: 1.10 src/objsvcs/transactions/ots_appc/otsaptnr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:40.929019384
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCPartner::recreateObject(OTSSRMLogSection*,CORBA::ULong):546
 reported an activity.
ExtendedMessage: The APPC Connection between local LU 'USIBMZP.PVT01001' and
 partner LU 'USIBMZP.CICS4' was recovered from the Transaction Service log.
RawDataLen: 0
-------------------------------
```

5. You can use the SOMCB_APPC_PARTNER_LIVES environment variable to instruct the transaction service to delete APPC Connections from the transaction log but make sure that these APPC Connections are not defined in the server's configuration *and* they do not have incomplete transactions.

When an APPC Connection is recovered from the transaction log, or when a PAA APPC request uses the APPC Connection for the first time, the Component Broker transaction service must exchange log names (XLN) with the tier-3 (partner SNA) system. This verifies that the transaction logs in both systems have not been deleted or changed since the last time they communicated with one another. If the transaction log has changed in either system, new PAA APPC requests are only

allowed to run if there are no partially-complete transactions involving the two systems. The following example shows a message stating that the *exchange log names* process between the Component Broker server and tier-3 system is fine.

**Note:** The *exchange log names* process for APPC is part of the SNA architecture and is required. It means that administrators of Component Broker servers should not delete the transaction service log files for servers using APPC unless all tier-3 systems that the server is communicating with are also being cold started. If the transaction service log for a Component Broker server is deleted and you see messages indicating that the tier-3 system will not exchange log names, then take one of the following actions:

- If the tier-3 system is CICS/ESA, issue the command CEMT SET CONNECTION(*xxxx*) NOTPENDING on the CICS region.
- If the tier-3 system is using an Encina PPC Gateway, use the command:

  ```
  ppcadmin cancel resync GatewayLUName CBLUName
  ```

  This command is documented in the *Transaction Series CICS Intercommunications Guide* under **ppcadmin**.

This action may damage the integrity of the data that the Component Broker server was updating in the tier-3 system.

```
_____________________________-
ComponentId: 262253
ProcessId: 505
ThreadId: 511
FunctionName: OTSAPPCPartner::setPartnerLogName(OTSAPPCLogName*,OTSAPPCLogName*)
ProbeId: 1420
SourceId: 1.10 src/objsvcs/transactions/ots_appc/otsaptnr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:42.116710367
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCPartner::setPartnerLogName(OTSAPPCLogName*,OTSAPPCLogName*)
:1420 reported an activity.
ExtendedMessage: Transactions are available between server CashAcctEMSvr (local LU name
 'USIBMZP.PVT01001') and partner SNA APPC system 'USIBMZP.CICS4'
RawDataLen: 0
_____________________________-
```

When log names have been exchanged, the server can send PAA APPC requests to the tier-3 system.

It is possible that a tier-3 system will send an *exchange log names* request to Component Broker from time to time. For the Component Broker to respond to this request, the application server has a thread dedicated to listening for incoming *exchange of log names* requests. When one is received, it is processed by another thread so that the listening thread can get back to monitoring for the next *exchange of log names* request.

The listening thread also monitors the status of the IBM Communication Server product. If it is running the **SNA Node** for the local machine, the following messages appear.

```
------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 513
FunctionName: OTSAPPCListenerNT::registerAsATM(OTSAPPCLUName*,CORBA::ULong)
ProbeId: 643
SourceId: 1.7 src/objsvcs/transactions/ots_appc/otsalist_nt.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:42.213633641
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCListenerNT::registerAsATM(OTSAPPCLUName*,CORBA::ULong):643
reported an activity.
ExtendedMessage: The SNA product 'IBM SNA' is running
RawDataLen: 0
------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 513
FunctionName: OTSAPPCListenerNT::startListening()
ProbeId: 379
SourceId: 1.7 src/objsvcs/transactions/ots_appc/otsalist_nt.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:06:42.217144746
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCListenerNT::startListening():379 reported an activity.
ExtendedMessage: Starting the SNA attach manager
RawDataLen: 0
------------------------------
```

If the following message appears, use the IBM Communication Server Node Operations application to start the node. See Checking the Status and Configuration of the SNA Network (page 91) for more information on how to do this.

```
------------------------------
ComponentId:    262253
ProcessId:      505
ThreadId:       513
FunctionName:   OTSAPPCListenerNT::registerAsATM(OTSAPPCLUName*,CORBA::ULong)
ProbeId:        657
SourceId:       1.7 src/objsvcs/transactions/ots_appc/otsalist_nt.cpp
Manufacturer:   IBM
Product:        Component Broker
Version:        2.0
```

```
SOMProcessType: 5
ServerName:     CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp:      10/30/98 12:07:13.923159238
UnitOfWork:     7905:littleapple.rchland.ibm.com
Severity:       3
Category:       2
FormatWarning: 0
PrimaryMessage: The function OTSAPPCListenerNT::registerAsATM(OTSAPPCLUName*,COR
BA::ULong):657 reported an activity.
ExtendedMessage: The SNA product 'IBM SNA' has stopped
RawDataLen:     0
-------------------------------
```

### An Unexpected Return code from IBM Communication Server

The Component Broker transaction service will write messages to the Component Broker's activity log if an unexpected result is returned by the IBM Communication Server. In most cases, a more descriptive message appears either just after this message, or earlier on in the life-time of the server to describe the problem. However, as this is a new function, the actual return values are also logged for problem determination purposes. For example, the following message reports an error in the ALLOCATE (page 170) call. The return value is in two parts: primary_rc and secondary_rc, which are documented in the "IBM Communication Server APPC Interface: Return Codes" on page 195.

```
-------------------------------
ComponentId:    262253
ProcessId:      746
ThreadId:       720
FunctionName:   OTSAPPCConvIdNT::setState(unsigned short,unsigned short,unsigned
long,OTSAPPCInterface::State)
ProbeId:        1234
SourceId:       %I% %W%
Manufacturer:   IBM
Product:        Component Broker
Version:        2.0
SOMProcessType: 5
ServerName:     myserver1
clientHostName:
clientUserId:
TimeStamp:      10/30/98 14:32:16.434741993
UnitOfWork:     8024:panda
Severity:       3
Category:       2
FormatWarning: 0
PrimaryMessage: The function OTSAPPCConvIdNT::setState(unsigned short,unsigned
short,unsigned long,OTSAPPCInterface::State):1234 reported an activity.
ExtendedMessage: The SNA library used for PAA APPC support returned '0x0300/0x04000000'
  from an APPC 'ALLOCATE' call.
RawDataLen:     0
-------------------------------
```

In the following entry, the call is nof_atm_wait_active and the return value is 61700 (=0xF104).

```
-------------------------------
ComponentId: 262253
ProcessId: 505
ThreadId: 513
FunctionName: OTSAPPCListenerNT::getInboundRequest(OTSAPPCInboundTPN**)
ProbeId: 575
SourceId: 1.7 src/objsvcs/transactions/ots_appc/otsalist_nt.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
```

```
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:07:12.017132256
UnitOfWork: 7905:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function OTSAPPCListenerNT::getInboundRequest(OTSAPPCInboundTPN**):575
  reported an activity.
ExtendedMessage: Unexpected error 61700 from the SNA call nof_atm_wait_active
RawDataLen: 0
-------------------------------
```

### Loading the Java Virtual Machine

The following entry appears in the activity log when the Java Virtual Machine (JVM) is loaded into a server. It shows the CLASSPATH and other attributes that it is using.

```
-------------------------------
ComponentId: 102
ProcessId: 505
ThreadId: 447
FunctionName: loadAndInitVM(JavaVM**,JNIEnv**,SOMException*)
ProbeId: 785
SourceId: 1.20 src/shasta/somrt/bossmerge/somshcb.C
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:10:22.837726560
UnitOfWork: 17481:littleapple.rchland.ibm.com
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage:
The function loadAndInitVM(JavaVM**,JNIEnv**,SOMException*):785 reported an
activity.
ExtendedMessage: Creating Java VM: CLASSPATH=D:\CBroker\ntApps\CashAccountEM\btc
aEMsobPO.jar;.;D:\CBroker\BIN;D:\CBroker\LIB\ivjeab.jar;D:\CBroker\LIB\hod20.jar
;D:\CBroker\LIB\ivjcicon.jar;D:\CBroker\LIB\sompart.zip;D:\CBroker\LIB\ivbtrolt.
jar;D:\CBroker\DATA\KEYRINGS;D:\CBroker\LIB\somojij.zip;D:\CBroker\lib\ivbtrutil
.jar;D:\CBroker\lib\ivbtrjrt.jar;D:\CBroker\lib\ivbtrc.jar;D:\CBroker\lib\ivbjdb
ug.jar;D:\CBroker\lib\ivbwin.jar;D:\CBroker\PARSER;D:\CBroker\LIB\IBMCBJS.ZIP;D:
\CBroker\LIB\somib01.zip;F:\Beans;D:\products\IBMVJava\Ide\program;D:\products\I
BMVJava\Eab\bin;d:\products\Cicscli\Java\JGate\classes;d:\products\jdk1.1.6\lib\
classes.zip;LIB\IBMCBJS.ZIP;LIB\IBMCBJS.ZIP;.;D:\CBroker\LIB\somshcl.zip;D:\CBro
ker\LIB\somshor.zip;;d:\products\jdk1.1.6\bin\..\classes;d:\products\jdk1.1.6\bi
n\..\lib\classes.zip;d:\products\jdk1.1.6\bin\..\lib\classes.jar;d:\products\jdk
1.1.6\bin\..\lib\rt.jar;d:\products\jdk1.1.6\bin\..\lib\i18n.jar, nativeStackSiz
e = 131072, java stack size= 409600, minHeapSize = 1048576, maxHeapSize = 167772
16, enableClassGC = 0, enableVerboseGC= 0
RawDataLen: 0
-------------------------------
```

### SECURITY ERROR Message on Each PAA APPC Request

If your Component Broker server does not have security enabled, then the following entry is written in the activity log whenever an APPC Connection is used. It means that no user ID or password will be sent with the PAA APPC request to the tier-3 system. The tier-3 system should have security set up to allow this.

```
------------------------------------
ComponentId: 262252
ProcessId: 505
ThreadId: 447
FunctionName:
ISecurityLocalObjectBasePrivateImpl_Credentials_Impl::get_mapped_security_info(
const char*,const char*,const CORBA::Any&)
ProbeId: 772
SourceId: 1.40 src/objsvcs/security/authn/ISecurityCredentialsI.cpp
Manufacturer: IBM
Product: Component Broker
Version: 2.0
SOMProcessType: 5
ServerName: CashAcctEMSvr
clientHostName:
clientUserId:
TimeStamp: 10/30/98 12:10:31.126985568
UnitOfWork: 17481:littleapple.rchland.ibm.com
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage:
The function ISecurityLocalObjectBasePrivateImpl_Credentials_Impl::get_mapped_
  security_info(const char*,const char*,const CORBA::Any&):772 reported an error.
ExtendedMessage: SECURITY ERROR: Security is not enabled for target CashAcctEMSvr.
RawDataLen: 0
------------------------------------
```

## Investigating Communications Server Problems

Communications Server provides the following applications to help you diagnose SNA communications problems:

- The Log Viewer application
- The Display SNA Sense Data application
- The Trace Facility application

Use these applications to perform these tasks:

- “Tracing Calls to the Communications Server” on page 89
- “Checking the Status and Configuration of the SNA Network” on page 91
- “Viewing Communications Server Messages” on page 95
- “Displaying Explanations of SNA Sense Data” on page 96
- Trace APPC Communications (page 61)

In addition, the local Component Broker application server and the remote tier-3 system may log explanatory messages when intersystem requests fail. For information about communication messages recorded by the Component Broker application server, see “Appendix H. APPC Messages” on page 263.

For an overview of CICS intersystem problem determination, see "Intersystem Problem Determination" in the *CICS Intercommunication Guide*.

**RELATED CONCEPTS**
IBM Communications Server for Windows NT (*System Administration Guide*)
Introduction to SNA (*System Administration Guide*)
Connections to Tier-3 Systems (*System Administration Guide*)

**RELATED TASKS**

"Troubleshooting APPC Problems" on page 77
Collect Information for Your SNA Configuration (*System Administration Guide*)
Configure Communications Server (*System Administration Guide*)
Configure an APPC Connection to a Tier-3 System for Use by Applications (*System Administration Guide*)

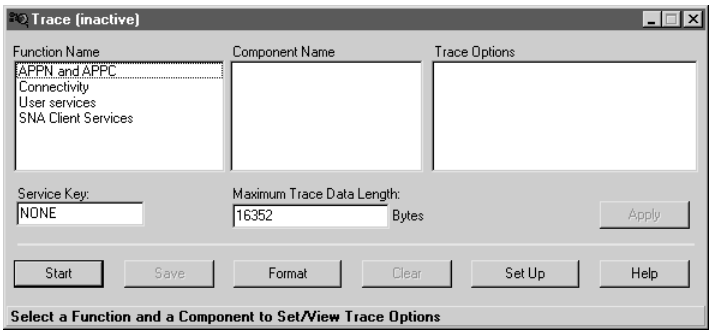
## Tracing Calls to the Communications Server

Use this procedure to trace calls made to Communications Server and the data that is flowing on the SNA network. You control traces by using the **Trace Facility** application provided with Communications Server. To trace calls made to Communications Server, complete the following steps:

1. Start the Communications Server Trace Facility application, by one of the following methods:
   - **Start - Programs - IBM Communications Server - Trace Facility**
   - From the **Launch** menu of the **SNA Node Operations** application.

   This displays the main Trace Facility window, as shown in the following figure:

   **The Trace Facility Application**

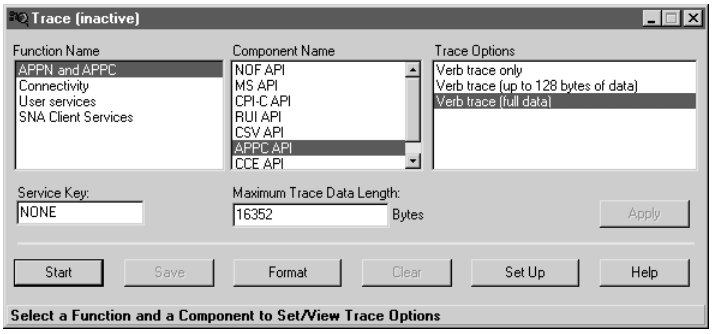



2. To trace the calls that an application server is making to Communications Server, select the **APPN and APPC** function name. This displays a list of components in the middle box.

   Component Broker application servers use the **APPC API** calls for intersystem requests and a few **NOF API** calls in the Local SNA listener calls when receiving inbound intersystem requests.

3. Select either of the **APPC API** or **NOF API** components, to display trace options that control the level of detail of trace to record, as shown in the following figure:

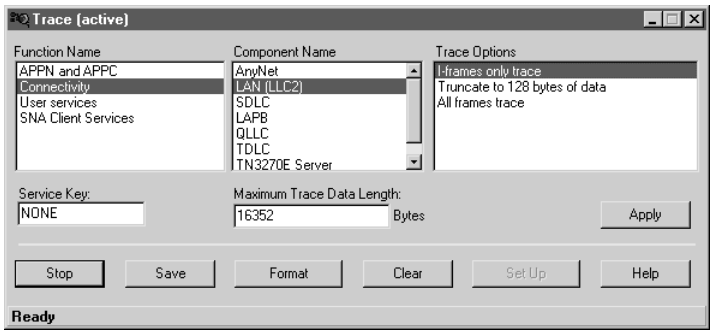
   **Selecting APPC Trace - Trace Inactive**




4. When you have selected the trace option you require, click the **Start** button.

The start button changes to a **Stop** button and the message **(active)** is displayed in the Title bar at the top of the window, as shown in the following figure:

**Selecting Connectivity Trace - Trace Active**

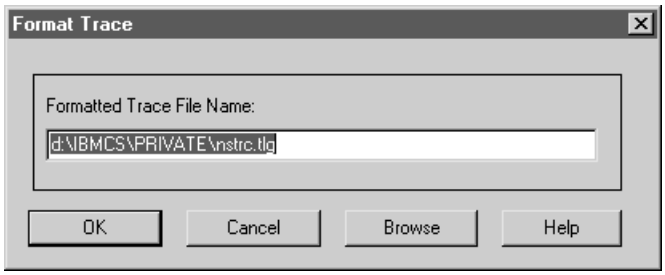



5. Optionally, you can select additional components to trace as before, and activate trace for those components by clicking the **Apply** button.

   In the example above, connectivity trace is selected to trace the data flowing to and from the local machine.

6. After you have run the request that you wished to trace, you should stop the trace (by selecting the **Stop** button), save it (by selecting the **Save** button), and format it (by selecting the **Format** button). You will be asked for a file name where the formatted trace should be written, as follows:

   **Formatting the Trace**




For more information about reading Communications Server traces, which requires some knowledge of the SNA architecture, see the SNA library.

**RELATED CONCEPTS**

IBM Communications Server for Windows NT (*System Administration Guide*)
Introduction to SNA (*System Administration Guide*)
Connections to Tier-3 Systems (*System Administration Guide*)

**RELATED REFERENCES**

"Appendix C. IBM Communication Server Trace Samples" on page 167

**RELATED TASKS**

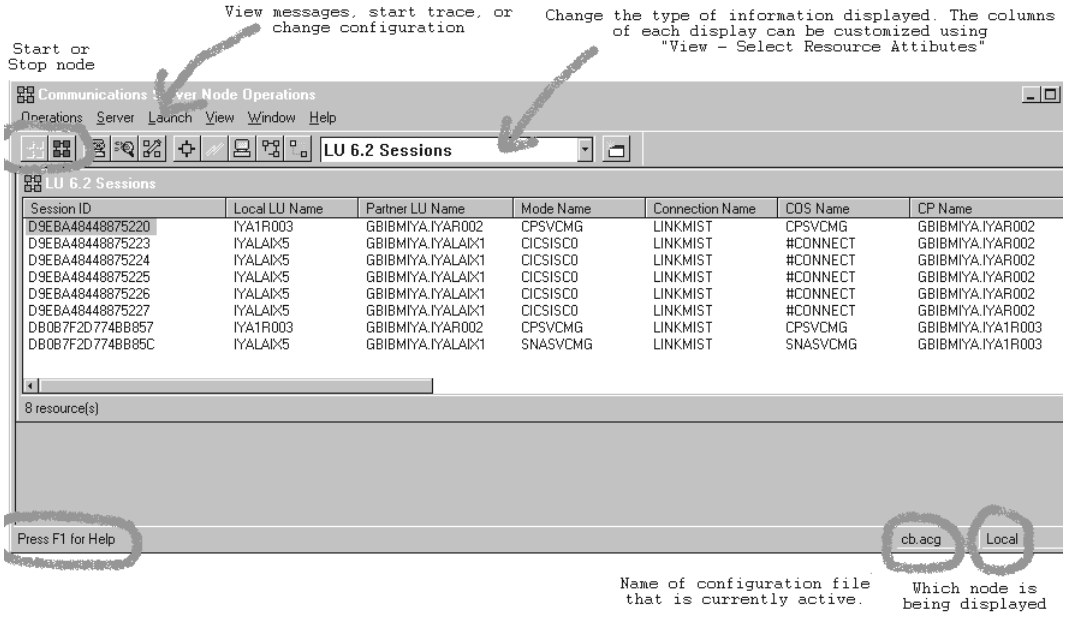
"Investigating Communications Server Problems" on page 88
Collect Information for Your SNA Configuration (*System Administration Guide*)
Configure Communications Server (*System Administration Guide*)
Configure an APPC Connection to a Tier-3 System for Use by Applications (*System Administration Guide*)

## Checking the Status and Configuration of the SNA Network

The **Node Operations** application (**pcsnops.exe**) is provided by IBM Communications Server to check the status of the SNA network and to detect any problems in the SNA configuration. It is used to:

- Start and Stop the Node (page 91) to view the SNA network.
- View Connections (page 92) to see which adjacent nodes are communicating with the local node.
- View Local LUs (page 93) to check the configuration of your Component Broker server's SNA name.
- View Modes (page 94) to see which tier-3 partner systems are communicating with your Component Broker server's local LU and how many session are active.
- View Active Sessions (page 95).

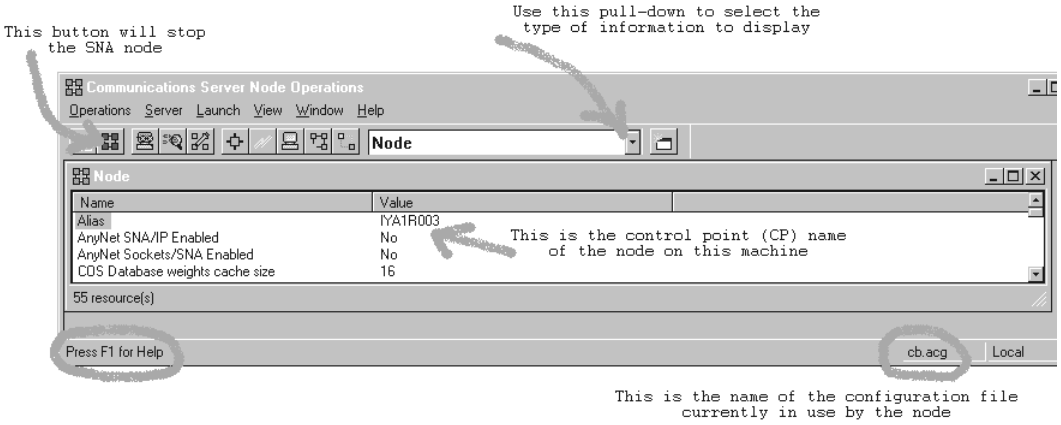
The screen below highlights the general features of the Node Operations application.




### Starting and Stopping the SNA Node

When you start the Node Operations application (pcsnops.exe) from IBM Communication Server when the SNA node is not running, the resulting window looks like the example below. Select the button with the four green squares on it to start the node. A dialog box then requests that you select the name of your SNA configuration file and it uses these SNA definitions to start the node.

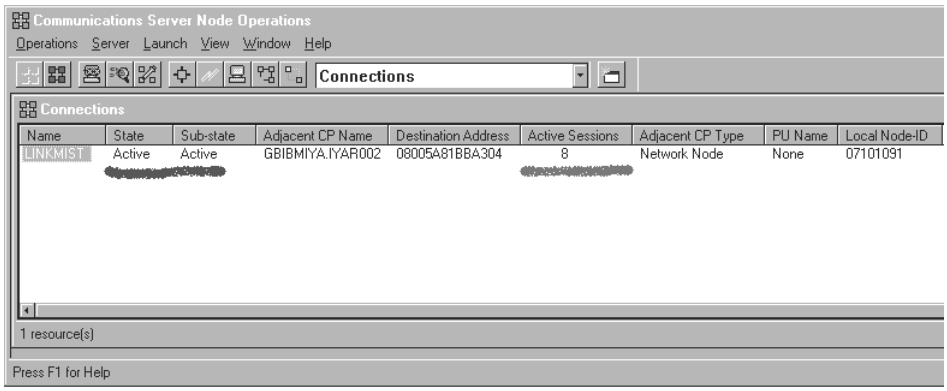

When the node is started, you can use the (white) pull-down menu to select the type of information to display. To stop the node, press the button with the four red squares on it and then select **Yes**. This will disconnect the machine from the SNA network.



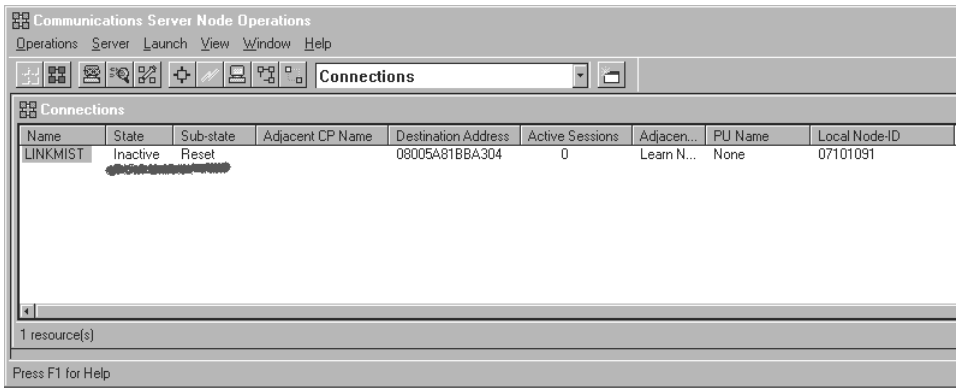

Note: it is also possible to start and stop the SNA node using the **csstart** and **csstop** commands. These are useful if you wish to create scripts to automatically control IBM Communication Server. The **csstart** command will activate the default configuration if the node is currently stopped. If the node is already running, it does nothing. The **csstop** command shuts down a running node. If the Node Operations application is running when these commands are issued, you have to select **Refresh** from the **View** pull-down menu to update the Node Operations information.

### Viewing Connections

When you start to investigate why the Component Broker server cannot communication with a tier-3 system, use the Node Operations application to view Connections. The following window shows an active connection running. The state is **Active** and sessions have started.




The next example is not so healthy as the state of the connection is **Inactive - Reset**. To start the connection, select the connection name (LINKMIST in the example) using a right mouse click. This displays a pop-up menu. Select **Start**. If the connection switches to **Pending** and then **Active** after a minute or so, then all is well. Re-run the PAA test and if it still fails, check the Modes (page 94). If the connection switches to **Pending** and then hangs, check that the connection/link is ready in the tier-3 system. If the connection switches to **Pending** and then back to **Reset** then either the information in the connection definition is incorrect (usually the network address) or SNA on the tier-3 system is not running.

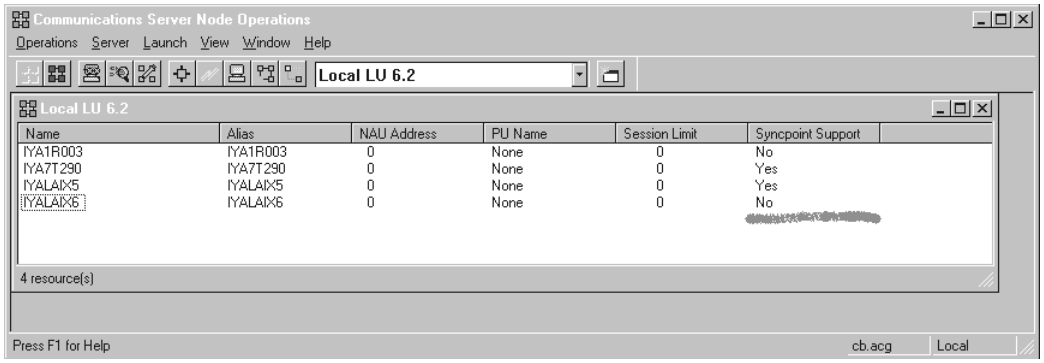



If no connections are displayed by Node Operations, check that the SNA node is started with a configuration file that contains the SNA connection definitions (*System Administration Guide*) in it.

### Viewing Local LUs

Use the Node Operations application to view Local LU 6.2 definitions and verify that the definition for each of the local LU names defined in Component Broker APPC Connections which is activated in the server is correct. Local LU names used by Component Broker should *not* be defined in your SNA configuration. This should be left to the Component Broker server when it first comes to use the local LU name. Component Broker needs to define its local LU names with **Syncpoint Support=yes** which is not possible to do using the node configuration application.

For example, the window below shows 4 local LUs.



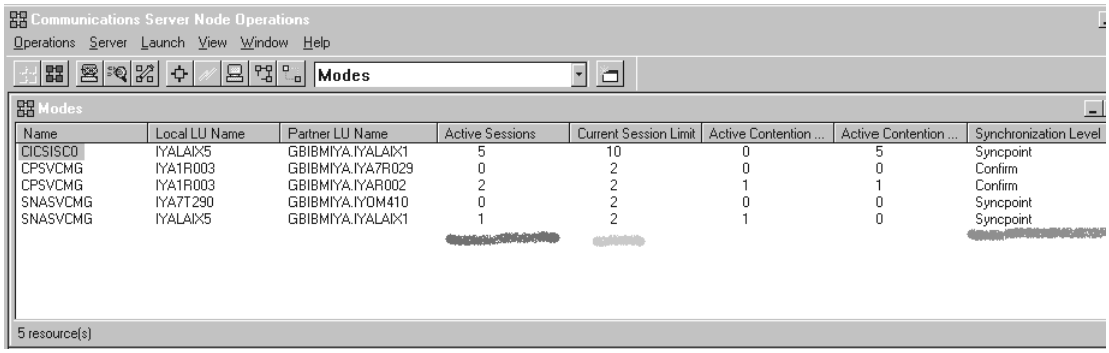

The first local LU is always the Control Point (CP) name of the SNA node. This should *not* be used by a Component Broker server. Of the remaining three local LUs, IYA7T290 and IYALAIX5 have **Syncpoint Support=yes** so they were correctly defined by a Component Broker Server. IYALAIX6, however, has **Syncpoint Support=no**. If a Component Broker server attempts to use it, the following message will appear in the activity log when the server first attempts to use the local LU:

```
ComponentId:    262253
ProcessId:      538
ThreadId:       629
FunctionName:   OTSAPPCLibraryNT::checkLocalLUName(OTSAPPCLUName*)
ProbeId:        847
SourceId:       1.6 src/objsvcs/transactions/ots_appc/otsalib_nt.cpp
Manufacturer:   IBM
Product:        Component Broker
Version:        2.0
SOMProcessType: 5
ServerName:     myserver1
clientHostName:
clientUserId:
TimeStamp:      10/30/98 18:43:38.349913043
UnitOfWork:     29748:panda
Severity:       3
Category:       2
FormatWarning: 0
PrimaryMessage: The function OTSAPPCLibraryNT::checkLocalLUName(OTSAPPCLUName*):847
 reported an activity.
ExtendedMessage: Local LU name 'GBIBMIYA.IYALAIX6' is defined in IBM Communication
Server with the syncpoint support disabled.  Please delete this local LU name
from the Communication Server configuration and recycle (stop then start) the
SNA node using the Communication Server Node Operations application.  Component
Broker will then define the local LU to Communication Server with the correct
settings when the local LU name is next referenced by the server.
```

Subsequent requests will get `0100/12000000` from an ALLOCATE request.

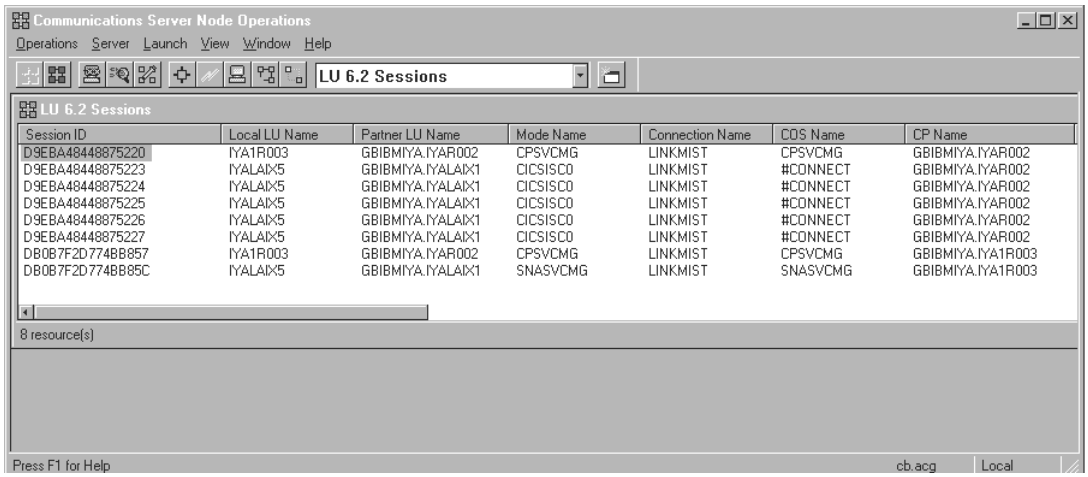
**Viewing Modes**

This panel allows you to see which tier-3 partner systems are communicating with your application server's local LU and how many sessions are active. It also displays other characteristics of the sessions; for example, the current session limit and synchronization level for the mode.




### Viewing Active Sessions

The following panel shows individual sessions that are bound between all local LUs and all partner LUs. You can unbind an individual session by selecting the session ID using a right-mouse button click which displays a menu with two **stop** options: **normal** and **abnormal**.




**RELATED REFERENCES**

"Investigating Communications Server Problems" on page 88
"Tracing Calls to the Communications Server" on page 89
"Troubleshooting APPC Problems" on page 77
"Appendix F. SNA Data Formats" on page 207

## Viewing Communications Server Messages

Use this procedure to view communications messages written by Communications Server to the PCWMSG.MLG log file.

The PCWMSG.MLG log file is located in the directory that was created during the install of Communications Server; by default, C:\Communications.

You can use the **Log Viewer** application provided with Communications Server to view the messages in the PCWMSG.MLG log file.

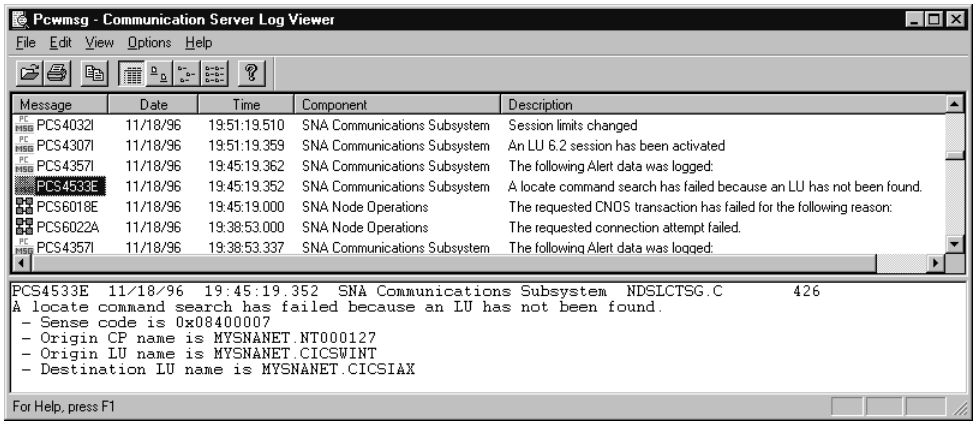
To view the messages written by Communications Server, complete the following steps:

1. Start the Log Viewer, by using one of the following methods:

- Click **Start - Programs - IBM Communications Server - Log Viewer**
- From the **Launch** menu of the **SNA Node Operations** application, click **Log Viewer**.

This displays the main window of the Log Viewer, as shown in the following figure. The messages are listed in the top part of the window, with some related information in the lower part of the window.

**The Log Viewer Application**




2. Optionally, to display a complete description of the Log Viewer, click **Help - Using Log Viewer** from the menu bar.
3. To display more information about a particular message number, complete the following steps:
   a. Select the message number in the top part of the main window. This highlights the message number, and displays more information about that message in the bottom part of the window.
   b. If you double-click on the message number, a *help* window for the message will be displayed.
4. The Communications Server messages often contain a SNA sense code in the form of 0xnnnnnnnn; for example, 0x08570003. Optionally, to display a full explanation of sense codes, use the Display SNA Sense Data Application provided with Communications Server.

**RELATED CONCEPTS**
IBM Communications Server for Windows NT (*System Administration Guide*)
Introduction to SNA (*System Administration Guide*)
Connections to Tier-3 Systems (*System Administration Guide*)

**RELATED TASKS**
“Displaying Explanations of SNA Sense Data”
“Investigating Communications Server Problems” on page 88
Collect Information for Your SNA Configuration (*System Administration Guide*)
Configure Communications Server (*System Administration Guide*)

## Displaying Explanations of SNA Sense Data

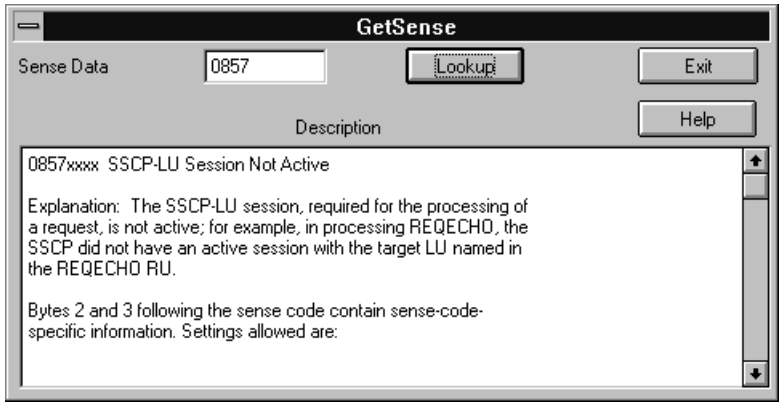
The Communications Server messages often contain an SNA sense code in the form of 0x*nnnnnnnn*; for example, 0x08570003. You can use the Display SNA Sense Data Application provided with Communications Server to get a full explanation of these sense codes.

To display a full explanation of an SNA sense code, complete the following steps:

1. Start the Display SNA Sense Data Application, by clicking **Start - Programs - IBM Communication Server - Display SNA Sense Data**.

   This displays the Display SNA Sense Data panel, as shown in the following figure:

   **SNA Sense Data Panel**




**RELATED CONCEPTS**

IBM Communications Server for Windows NT (*System Administration Guide*)
Introduction to SNA (*System Administration Guide*)
Connections to Tier-3 Systems (*System Administration Guide*)

**RELATED TASKS**

"Investigating Communications Server Problems" on page 88

## Checking the States of Resources in a CICS Region

The CICS Enhanced Master Terminal (CEMT) application allows you to query the states of different resources in the CICS region. The following CEMT commands are useful when investigating PAA APPC problems.

- CEMT INQUIRE TRANSACTION (page 97)
- CEMT INQUIRE PROGRAM (page 98)
- CEMT INQUIRE CONNECTION (page 98)
- CEMT INQUIRE MODE (page 100)
- CEMT INQUIRE TASK (page 100)

For more information about these commands, please see the documentation supplied with your CICS system.

**CEMT INQUIRE TRANSACTION**

CEMT INQUIRE TRANSACTION is useful for verifying that the transaction program name in the Component Broker APPC Connection exists on the tier-3 system.

The CEMT screen below shows the CICS transaction program names that begin with the letter "B" which are available in the CICS region. The command used to display this panel is **CEMT I TRAN(B*)**.

```
I TRAN(B*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Tra(BBCA) Pri( 001 ) Pro(BBCASHAC) Tcl( DFHTCL00 ) Ena
 Tra(BBCU) Pri( 001 ) Pro(BBCITYUT) Tcl( DFHTCL00 ) Ena
 Tra(BBPE) Pri( 001 ) Pro(BBPERSON) Tcl( DFHTCL00 ) Ena
 Tra(BDPL) Pri( 001 ) Pro(CICSDPL ) Tcl( DFHTCL00 ) Ena
 Tra(BDTP) Pri( 001 ) Pro(CICSDTP ) Tcl( DFHTCL00 ) Ena
 Tra(BOSS) Pri( 001 ) Pro(BBMENU  ) Tcl( DFHTCL00 ) Ena
 Tra(BRWS) Pri( 001 ) Pro(DFHPBRW) Tcl(  DFHTCL00 ) Ena
                                                SYSID=CICT APPLID=CICSTR
  RESPONSE: NORMAL                        TIME:  09.25.12  DATE: 07.29.98
PF 1 HELP       3 END                 7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

**CEMT INQUIRE PROGRAM**

The CEMT INQUIRE PROGRAM command is useful for verifying that the real program pointed to by the transaction definition for the transaction program name in the Component Broker APPC Connection exists on the tier-3 system.

The CEMT screen below shows the CICS programs that begin with the letters "CICS" which are available in the CICS region. The command used to display this panel is **CEMT I PROG(CICS*)**.

```
I PROG (CICS*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Prog(CICSDPL ) Len(0009432) Cob Pro Ena Pri     Ced
    Res(001) Use(0000000440) Bel Uex Ful
 Prog(CICSDTP ) Len(0000000) Cob Pro Ena Pri     Ced
    Res(000) Use(0000000000) Bel Uex Ful
                                                SYSID=CICT APPLID=CICSTR
  RESPONSE: NORMAL                        TIME:  05.00.23  DATE: 07.29.98
PF 1 HELP       3 END                 7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

**CEMT INQUIRE CONNECTION**

CEMT INQUIRE CONNECTION is useful for checking the state of connections to other systems (including Component Broker application servers) that the CICS region is communicating with.

The CEMT screen below shows the state of *all* connections to other systems that the CICS region is communicating with. The command used to display this panel is **CEMT I CONN**.

```
 I CONN
 STATUS:  RESULTS - OVERTYPE TO MODIFY
+ Con(CB11) Net(CBS11001)      Ins Rel          Vta Appc
  Con(CB12) Net(CBS12001)      Ins Rel          Vta Appc
  Con(PA01) Net(PAA01001)      Ins Rel          Vta Appc
  Con(PA02) Net(PAA02001)      Ins Rel          Vta Appc
  Con(PA03) Net(PAA03001)      Ins Rel          Vta Appc
  Con(PA04) Net(PAA04001)      Ins Acq      Xno Vta Appc
  Con(PA05) Net(PAA05001) Pen  Ins Rel          Vta Appc
  Con(PA06) Net(PAA06001)      Ins Rel          Vta Appc
  Con(PA07) Net(PAA07001)      Ins Rel          Vta Appc
  Con(PA08) Net(PAA08001)      Ins Acq     Xok  Vta Appc
  Con(PA09) Net(PAA09001)      Ins Rel          Vta Appc
  Con(PA10) Net(PAA10001)      Ins Rel          Vta Appc
  Con(PA11) Net(PAA11001) Pen  Ins Rel          Vta Appc
  Con(PA12) Net(PAA12001)      Ins Rel          Vta Appc
  Con(PV01) Net(PVT01001)      Ins Acq     Xok  Vta Appc
  Con(PV02) Net(PVT02001)      Ins Rel          Vta Appc
  Con(PV03) Net(PVT03001)      Ins Rel          Vta Appc
+ Con(PV04) Net(PVT04001)      Ins Rel          Vta Appc

                                                SYSID=CICT APPLID=CICSTR
  RESPONSE: NORMAL                               TIME:  04.55.16  DATE: 07.29.98
PF 1 HELP       3 END                    7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

The table below describes the states for a connection:

| | |
|---|---|
| Out Rel | **OutOfService Released**<br>This connection is not available for use. Attempts to use this connection will fail. To make the connection available, type **ins** over **Out** on the CEMT connection entry and press Enter. You should see the connection switch to **Ins Rel**. |
| Ins Rel | **InService Released**<br>This connection is available for use but not currently connected to the partner system. To start the connection, type **acq** over **Rel**. If the partner system is running and the SNA definitions are correct, you should see the connection switch to **Ins Acq Xok**. |
| Pen Ins Rel | **InService Released, incomplete transactions PENding**<br>This connection is available for use but not currently connected to the partner system. In addition, CICS has some transaction recovery work to do when the connection is next started. To start the connection, type **acq** over **Rel**. If the partner system is running and the SNA definitions are correct, you should see the connection switch to **Ins Acq Xok**. |
| Ins Acq Xno | **InService Acquired, eXchange log names NOt successful**<br>This usually means that the connection between CICS and the Component Broker application server was started (Acquired) when the application server was not running. Restart the application server, then rerun CEMT I CONN to see if it switches the state to **Ins Acq Xok**. If it does switch the state correctly, then try running a PAA APPC application to activate the Component Broker APPC Connection. If the state still has not changed, then look for error messages in the Component Broker application server's activity log and follow the instructions for any APPC messages found. If no messages appear, it is likely that the CICS region is connecting to a different LU name from that used by the Component Broker application server (check the CEMT I CONN display with the local LU name in the Component Broker APPC connection) or the local LU name for the Component Broker application server is defined in the SNA network as located on a different machine from where your Component Broker application server is located. |

| | |
|---|---|
| Pen Ins Acq Xno | **InService Acquired, eXchange log names NOt successful, incomplete transactions PENding**<br>This usually means that the connection between CICS and the Component Broker application server was started (Acquired) when the Component Broker application server was not running. In addition, the **Pen** status means there are incomplete transactions to recover when the Component Broker application server comes up. In order for recovery to be successful, the Component Broker application server needs to use the same transaction log it used before. Restart the Component Broker application server and monitor the messages written to the activity log, looking for error messages concerning the recovery of transactions. Follow the instructions for any APPC messages found. |
| Ins Acq Xok | **InService Acquired, eXchange log names OK**<br>This connection is available and can be used for PAA APPC requests. |

**CEMT INQUIRE MODE**

The CEMT INQUIRE MODE command is useful to view how many sessions are active for a particular mode name. The number of active sessions determines the number of concurrent APPC requests can run as each session can only run one request at a time. The information displayed here can also be seen using the Node Operations application as described in IBM Communication Server Node Operations (page 94).

The CEMT screen below shows the SNA modes that are currently defined for each connection to the CICS region. The command used to display this panel is **CEMT I MODE**.

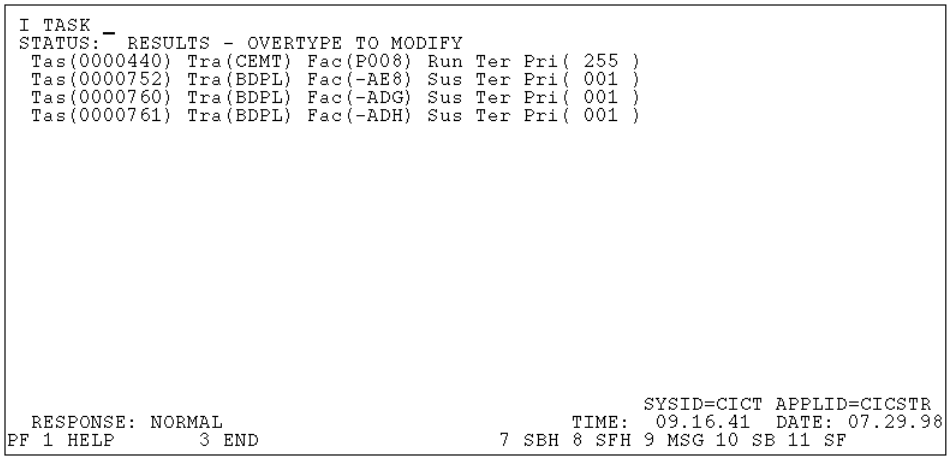
```
 I MODE
 STATUS:  RESULTS - OVERTYPE TO MODIFY
+ Mod(SNASVCMG) Con(PA08) Max(002) Ava( 002 ) Act(001)
  Mod(LU62PS  ) Con(PA08) Max(004) Ava( 004 ) Act(002)
  Mod(SNASVCMG) Con(PA09) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PA09) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PA10) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PA10) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PA11) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PA11) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PA12) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PA12) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PV01) Max(002) Ava( 002 ) Act(001)
  Mod(LU62PS  ) Con(PV01) Max(004) Ava( 004 ) Act(004)
  Mod(SNASVCMG) Con(PV02) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PV02) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PV03) Max(002) Ava( 000 ) Act(000)
  Mod(LU62PS  ) Con(PV03) Max(004) Ava( 000 ) Act(000)
  Mod(SNASVCMG) Con(PV04) Max(002) Ava( 000 ) Act(000)
+ Mod(LU62PS  ) Con(PV04) Max(004) Ava( 000 ) Act(000)

                                                SYSID=CICT APPLID=CICSTR
  RESPONSE: NORMAL                          TIME:  09.20.37  DATE: 07.29.98
PF 1 HELP       3 END                  7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

**CEMT INQUIRE TASK**

The CEMT INQUIRE TASK command is useful to view one or more CICS tasks that are currently running in the CICS region.

The CEMT screen below shows *all* the CICS tasks that are currently running in the CICS region. The command used to display this panel is **CEMT I TASK**.




**RELATED REFERENCES**

"Troubleshooting APPC Problems" on page 77
"Appendix H. APPC Messages" on page 263

## Troubleshooting Dr. Watson Errors

WIN When a Dr. Watson window appears while running a Component Broker application, generally means that some system code, or some code that the system is dependent on, has taken a hard fault or crash. When this occurs, click **ok** in the window. Proceed to the directory where Windows NT is installed (for example, C:\WINNT) and capture the relevant entry or entries from the drwtsn32.log. You can edit the file and search for **pid=**. A sample entry header looks like this:

```
Application exception occurred:
App: (pid=465)
When: 1/27/1998 @ 0:2:36.731
Exception number: c0000005 (access violation)
```

Refer to the Windows documentation for more information on what you can do when you get a Dr. Watson error.

**RELATED REFERENCES**

"Chapter 16. Troubleshooting Component Broker Run-Time Problems" on page 71

# Chapter 17. Problem Determination Hints and Tips

When you encounter a Component Broker run-time error, you can follow some of these hints and tips to help you quickly gather relevant data to diagnose the problem. You can also consider implementing some of the Component Broker settings that are described here because they may prevent some run-time errors from occurring. The following hints and tips topics are covered:



## Hints and Tips: Activity Log

In most problem determination situations, you would read the entries in the activity log. It is, therefore, important to find ways to quickly pinpoint the log entries that pertain to the problem that you are investigating. One way to do this is to reduce the activity log to a more manageable size. Here are some ways to do that.

**Set the Size of the Activity Log**
Do *not* use **Activate** to start the servers. Start them one at a time. The size of the activity.log file can be set above its default (1024K or 1 MB) as follows:

1. Go into the System Manager user interface under **host images**, right click on the host image and select **Edit**.
2. Select the tab, **Log File Controls**, and change activity log maximum size to the desired number of K bytes. You may want to use 15K for testing robustness and 50K for long runs. This applies to each host that has Component Broker logging. **Note**: The activity log wraps after it is full.

**Format and Erase Activity Logs: Smaller Activity Logs**
Smaller activity logs may speed up your problem determination process. If the run-time error can be reproduced by rerunning your application, consider performing the following steps to create a set of smaller activity logs:

1. Format your last activity log into a file and save it. Delete the activity log. Rerunning the application with a new activity log will minimize the extraneous information in it.
2. Activate your configuration.
3. When the activation is complete, do a showlog to format the activity log to *log1* and delete the activity log.
4. Run your test. After the test, do a showlog to format the activity log to *log2*.

You now have a set of smaller formatted activity logs to work with. For example, if *log2* shows that a client could not find a factory, *log1* will show you why that factory was not registered in lifecycle. You can also consider giving better file names for the formatted activity logs, for example, you can specify the following showlog command:

```
showlog activity.log -d > serverStartup.980808.firstrun.out
```

**System Manager User Interface or showlog to Format the Activity Log**
You get identical information when you use the System Manager user interface to browse the activity log or use **showlog** (with **-debug**) to format the entries into an output file. You cannot use the System Manager user interface to dump the formatted contents of the activity log into an output file. You should run showlog on the host that created the activity log.

To browse the activity log using the System Manager user interface, follow these steps:

1. Expand the **Host Images** folder, expand the host image that you are interest in, and then expand the **Activity Log Image**.
2. Right click on **Latest** and select **Browse** to view the activity log in a window.

**RELATED REFERENCES**
"Chapter 2. Activity Log for Problem Determination" on page 3
"Reading the Activity Log" on page 6
"showlog Utility" on page 68

## Merge cout Trace and ORB Communication Trace

You can perform more advanced debugging if you implement the **cout** tracing as described in "Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace" on page 51 together with the ORB Communications Trace. You may find it useful to view the output from both traces in a single log. This way, you can, at the same time, analyze the function calls and the GIOP packets that are sent by the ORB. To get the merged output, direct the ORB Communications trace to the console log.

You must start the server in a command window and follow these steps to capture all the tracing information in the console log:

1. Make sure that the ORB Communications trace is not set for the server image.
   a. Display the System Manager user interface, and set the user level to **Expert**.
   b. Expand your **Host Images**.
   c. Expand the **Server Images** folder.
   d. Left click on your server image to see if it is running. The status bar at the bottom of the System Manager user interface displays the state (and health) of the selected server.
   e. If your server is running, right click on your server image and select **Stop** to stop the server.
   f. Right click on your server image and select **Edit**. This displays the Object Editor for the Server Image.
   g. In the Object Editor window, click the **Component Trace** tab.
   h. If the **ORB Communications trace level** attribute value is *not* **None**, then select **None** to disable the trace. Click **Apply** and **OK** to apply the change.
2. In a command window, run the server application as follows:
   a. Set the environment variables:
      - SOMCBENV=s:*serverImageName*
      - HOSTNAME=*fullyQualifiedHostName*
      - SOMDGETENV=1

- SOMDCOMDEBUGFLAG=257
  This turns on the ORB Communications trace.

b. Run the server by typing:

somsrsm > *outputFile*

You will get all the trace information in a readable format interleaved in *outputFile*.

**RELATED REFERENCES**

"Chapter 11. ORB Communication Trace" on page 41
"Chapter 12. Tracing Function Calls in the Generated Code: XYZ Trace" on page 51

## Use Standalone Servers

There is no advantage to using server groups unless you plan to implement Workload Management. Server group configuration is more complex than that of a standalone server. If you have a server group and want to convert it to a standalone server, do the following:

1. Go to the configuration, under **Server Groups—>Servers**, right click on the server that you want to convert.
2. Select **Convert to Free Standing**. You will see the exact information and steps required to do the conversion.

**RELATED REFERENCES**

"Chapter 17. Problem Determination Hints and Tips" on page 103

## Run-Time Environment Settings

You can change run-time environment settings in the image world. This is done after each activation so you may not want to use **Activate** to start *all* the servers. You can start servers one at a time and control the settings that are described here.

**Log Sizes**

The size of the activity log can be set to a number that is greater than its default (1024K or 1 MB). See "Hints and Tips: Activity Log" on page 103 for instructions on how to do this. You can use 15MB for testing robustness and 50MB for long runs. This applies to each host that has Component Broker logging. The activity log wraps after it is full.

**Console Log**

The cout or println output from your programs can be piped into a file (console log). There is a console log for each server. To control this, you can use the System Manager user interface to do the following:

1. Under **server images**, right click on the server image that you want to modify and select **Edit**.
2. Tab to **Log Controls** and select **Console Disposition** to **file**.
3. Specify the name of the file for the console output. It will be written to a directory named %SOMCBASE%\service\server*servername*. Even though each server has its own directory, name the file something like *servername*.console.log so that later, when all console logs are gathered into a specific place, console logs from various servers will be distinguishable.

4. After this change, the server will not start. It will have to be **Enabled** from the server image to start.

**DB2 Preparation**
Go to the database server machine to make these modifications:

1. Setup the diagnostic log by going to the Control Center.
2. Under the instance name, right click and select **Configure**. From there, pick the log **Diagnostic** tab. Under this tab, select **Diagnostic error capture level** and **All errors, warnings, and informational messages**. Press Enter to make the appropriate changes. You will be told to restart DB2.

**DB2 Heap Size**
You can use the DB2 Control Center to increase your application heap size (for example, from 128 to 256) to eliminate the DB2 failure error that is caused by not having enough storage in the application heap. Here are the steps that are needed to configure the DB2 database from the Control Center:

1. Bring up the Control Center. WIN On Windows NT select **Start**->**Programs**->**DB2 for Windows NT** -> **Administration Tools** -> **Control Center**.
2. Find the Database that you want to reconfigure. Select **Open up system name** -> **Instances** -> **Node name** -> **Databases**.
3. Select the Database Name and right click to select **Configure**. This brings up the Configure Database GUI.
4. Click on the **Performance** tab.
5. In the performance tab there are different parameters that can be configured. There are several different heap size entries and you can specify the Application Heap Size.

You can also use the Performance Configuration SmartGuide to help you determine the heap size. To invoke the SmartGuide, instead of clicking on **Configure**, choose **Configure performance**.

**Paging File Size**
WIN Consider increasing the paging file size (for example, to 400 or 500MB) to reduce the possibility of server activation failures or poor system performance.

**DNS IP Address**
Verify if the DNS IP address is incorrectly set because it can cause DCE to be very slow.

**Timeouts on the Server Hosts**
Most of the default timeout settings in the Component Broker are set up to enable single-machine debugging. This section describes each of the relevant timeout values and what criteria should be used in selecting the proper value in the context of a multi-host environment.

The following descriptions focus on altering timeouts in the model world of Systems Management. That means that an activation of the proper configuration must be done for these timeouts to take effect. It also implies that additional Server Hosts that are configured with servers will have similar timeouts. If timeout modifications are only a tactical testing statement, the changes described below can all be made in the image world as well.

There are three groups of major timeouts to consider on server hosts:

- Name Server Request Timeout (page 107)

- Application Server Request Timeout (page 107)
- Daemon Timeouts (page 107)

**Name Server Request Timeout**
Under the configuration being used, look for **Name Servers**. There should be a **Sample Name Server**. This is probably the server model that is used when a name server is created. Right click on this server to select **Edit**, and tab to the **Orb** tab. The **request timeout value** (default 30) represents the amount of time that a Name Server should wait when requesting something from a remote server before it times out. Change this entry to a value between 180 and 300 seconds when working in a multi-host environment. This value is seldom used because Name Servers seldom make remote requests.

**Application Server Request Timeout**
Under **Servers (free standing**) or **Server Groups**, look for **Servers (member of group)**. This is probably the server model that is used when an application server is created. Right click on this to select **Edit**, and tab to the **Orb** tab. The request timeout value (default 30) represents the amount of time that an Application Server should wait when requesting something from a remote server before it times out. Specify a value between 180 and 300 seconds when working in a multi-host environment. This value can be used in multi-host, multi-server environments. If the servers are configured on multiple hosts, then the network latency is part of this value.

The sgcp server and sggw server may also be listed under the Server Group. These are used if Workload management (WLM) support is being leveraged. Although these servers only exist on one host in the Server Group, they do have a request timeout. The request timeout value (default 30) represents the amount of time that these servers should wait when requesting something from a remote server before it times out. Specify a value between 180 and 300 seconds when working in a multi-host environment. If the servers are configured on multiple hosts, then the network latency is part of this value.

**Note**: The name server, sgcp server and sggw server are based on the same process model and infrastructure code as application servers. That is why they are similar.

Still under the configuration and under **Client Styles**, right click on the *hostname* **Default Client** to select **Edit**. Select the **Orb** tab to see the **request timeout value**. This represents the amount of time that this host, when it is used as a client (for example, SOMCBENV=c:*hostname* Default Client), should wait for a request to a server. This is only used if you are running from a command window on the server host that has the proper environment setting. This is not used when servers call other servers (described above). Set this value based on how long you are willing to wait for a server to respond. Error handling in client programs and general client application design must be considered when setting this value.

**Daemon Timeouts**
Under each Host Image, there is a folder entitled **Daemon Images**. This contains the settings for the somorbd process which serves as the ORB daemon. The **Daemon Image** has a **request timeout** and it should be set similar to the application server settings. It also has a **registration timeout** which represents the amount of time that the daemon should wait for a server that is being started by the daemon, to register as being active and ready to accept work. A runOnRequest

server that is activated by the daemon would have to start within this time period. Specify a value between 180 and 300 seconds.

**Timeouts on the Client Hosts**
Managed clients have a **Client Style** and **Agent Image** and timeout values should be set according to the guidelines described in this section. This applies to C++ clients in the Component Broker environment.

**Timeout Synchronization and Interactions**
If the request timeout is set to 30 seconds and a transactions timeout is set to 180, then there are 150 seconds when resources might be locked and a retry might not work. This assumes that a single server is involved. A request timeout of 90 seconds on a client and 30 seconds from server 1 to server 2 means something different. The clock on the 90 second timeout runs throughout the request. The clock on the server to server request timeout runs only while a server is making a request on another server. There are some simple rules:

1. Of all the relevant request timeouts, the client request timeout should be the largest and always exceed the total expected server request timeout time.
2. If a server talks to two other servers each with 60 second timeouts, then the client request timeout should be at least 120 seconds and, more likely, at least 150 seconds to allow 30 seconds of actual work to be done on the first server that it is talking to. The Name Server is the same (from the timeout perspective) as any other application server.

**PATH Setting for Java Applications**
The JDK directory (for example, JDK1.1.6) should be in front of the PATH system environment variable to avoid conflicts with other products that may contain Java DLLs (such as Lotus products). These conflicts have been known to cause segmentation faults when performing an activate in System Management.

If you need to manually modify the system PATH environment variable to move your JDK directory, there is a known problem where the link between the system and user PATHs gets broken. To get around this problem, you should link them back up by putting %PATH% at the end of your user PATH variable.

**RELATED REFERENCES**
“Preparation Before Each Test Run”
“Merge cout Trace and ORB Communication Trace” on page 104

## Preparation Before Each Test Run

When you are testing your application, you should follow these steps prior to running your application so that you will have better problem determination data when you encounter an failure or error:

1. Clean the activity.log file. Go to the %SOMCBASE%/service directory and remove the activity.log file.
2. Cleanup the Database Environment. If you are using DB2, go to a DB2 Command Prompt and do the following:
   a. Ensure that for each database you get nothing when you enter **list indoubt transactions**. If there are some entries in the list, use **list indoubt transactions** with prompting to forget or roll back the transactions.

b. There should be no extra connections when you issue **list applications**. If there are some entries in the list, **force applications all** to clean up the connections.

c. There should be no suspicious locks when you issue **get snapshot for locks** on the database name.

d. Clean up the database table that you are going to use to return it to the proper state in terms of how many rows there are and what the test data should be.

3. Cleanup the console logs. For each server, remove the console logs.
4. Remove the OTS Logs. These are usually located in the %SOMCBASE%/data/otsLog directory.

**RELATED REFERENCES**

"Run-Time Environment Settings" on page 105
"Running the Test"

## Running the Test

**During Startup**

Start the servers in order and write down PID numbers and memory utilization figures.

**Getting the PIDs**

It is always useful to have the processIDs of the servers that are involved in tests. While this information appears in the logs, there is additional value in writing down these numbers during the test run. You may want to refer to these processIDs when you are reading the activity log. To get the processID, follow these steps:

1. In the System Manager user interface, go to the **Host Images** for the host and then to the servers folder. Under the **host image**, expand the servers folder, right click on the server and tab over to the **Private** tab.
2. Look at the **process identifier** for the processID.

This procedure works for all servers including sgcp, sggw, Name Servers, and Application Servers.

**Get Snapshots of the Activity Log**

You can optionally get snapshots of the activity log. Sometimes it is easier to find relevant entries if the log is in smaller chunks. If there is a recreatable problem, erase the old activity log. After doing a successful startup, take a snapshot of the activity log. To take a snapshot of the activity log, follow these steps:

1. Run the **showlog** command on the activity log.

```
cd %SOMCBASE%\service
showlog activity.log -d > serverStartup.980808.firstrun.out
```

2. Delete the activity.log file. A new activity log will be created with the next entry. The activity log is not *open* so you can take a snapshot of it while the servers are running.

**Note**: Console logs are *open* when servers are running. You have to stop the servers before you can copy them to another directory.

### Monitor System Statistics

You may want to look at memory usage (virtual memory) of key processes by using these commands:

1. somsrsm
2. somorbd
3. bgmain

If any of these displayed memory usage figures are growing after a test reaches steady state, then there is a memory leak somewhere. The memory usage figures should also reach a steady state during the course of an extended run. You can also look at threads and handles on the tasks manager.

**RELATED REFERENCES**

“Run-Time Environment Settings” on page 105
“Preparation Before Each Test Run” on page 108

# Chapter 18. Report a Problem to IBM

Use the information in this topic to help you report a suspected problem to IBM.

Before reporting problems, please review the known restrictions in the Late Breaking News. If the problem is not covered by the known restrictions, please contact your IBM sponsor to communicate the problem to the Component Broker development organization. Alternatively, you can submit a problem report directly through the Component Broker main Web page (http://www.software.ibm.com/ad/cb) by selecting the Support icon.

When preparing to report a problem, consider the information in the following topics, which will help you gather the appropriate diagnostic information and package it. This in turn, will help the support team to provide you with the appropriate service.

- Required Information (page 111)
- Package the Information (page 113)

**Required Information**
When preparing to report a problem, gather the information that is listed in this topic. This information will help to diagnose the problem.

**Note:** This information assumes that you used the default values for the various logs. If you modified these default values, please provide the corresponding files and directories in the package.

- WIN When you report a problem, the following information is helpful in problem diagnosis.
  - The environment variables
  - The registry settings
  - The C:\dbg.out file
  - The x:\CBroker\service directory
  - The x:\CBroker\data directory
  - The DB2 diagnostic log
  - The DCE logs (if the problem is DCE related)
  - The Model Consistency Checker Output
  - The problematic application
- AIX When you report a problem, the following information is helpful in problem diagnosis.
  - The environment variables
  - The smit.log file
  - The /var/CBConnector/service directory
  - The /var/CBConnector/data directory
  - The DB2 diagnostic log
  - The DCE logs (if the problem is DCE related)
  - The Model Consistency Checker Output
  - The problematic application

For more information, see the following descriptions:

**Environment Variables** WIN

Open a Windows NT command window and get a copy of the environment variables for your environment. To create this information, run the following command:

```
set > setenv.out
```

This command places a copy of your current environment variables in the setenv.out file.

**Environment Variables** AIX

A copy of the environment variables are in the file CBConnector.profile. A copy of this file is placed in the $HOME directory.

**Registry Settings** WIN

Component Broker saves supporting information in the NT Registry. To create a copy of this information from the NT Registry, do the following:

1. Type **regedit** on the command line to open the Registry Editor.
2. Expand **HKEY_LOCAL_MACHINE—>SOFTWARE—>IBM** folders to select **ComponentBroker** Registry Key.
3. From the menu bar, select **Registry—>Export Registry File**.
4. Select the folder for the output file. Fill in the **File name** with CBRegistry.out and click **Save**.

This places a copy of your NT Registry settings in the CBRegistry.out file.

**dbg.out** WIN **or smit.log** AIX

This file is created automatically for you during Component Broker installation. It contains information pertaining to the various install and uninstall steps. This file is cumulative of all Component Broker install and uninstall (regardless of success).

**Service Directory**

This directory contains the activity log and subdirectories for any trace logs that may have been generated. This directory with its subdirectories are useful in problem diagnosis.

**Data Directory**

This directory and its subdirectories contain much of the required data for the system management, transactions, and Interface Repository components.

**DB2 Diagnostic Log**

This log helps to decipher DB2 related situations. This log is in your DB2 instance directory.

- WIN The default location is x:\CBroker\sqllib\db2\db2diag.log.
- AIX The default location is $HOME/sqllib/db2dump/db2diag.log.

**DCE Logs**

These logs help to resolve DCE related problems. The files include:

- %DCELOC%\dcelocal\dcestart.log
- The files in the DCE directory %DCELOC%\dcelocal\var\svc.

**Model Consistency Checker**

Run the Model Consistency Checker to report on the state of the model and save the output. Turn on all the optional checking when running this checker. For more information on how to do this, refer to "Check a Model for Consistency" in the *Application Development Tools Guide*.

**Problematic Application**

If the problem goes beyond the basic Component Broker functions (that is, the problem can only be re-created using the client application), it may be useful to have a copy of this application package. The best way to provide this information is to build a disk image. This install image should include the DDL, DLL, shared library, executable, and bind files (as applicable).

**Package the Information**

If possible, zipping or compressing all of these files and directories into a single file (or, at least a single file for each information item in the list) is the easiest method of packaging this information. When sending this information to IBM, please provide the tool and options used to create the zipped or compressed files.

**RELATED REFERENCES**

"Chapter 1. Problem Determination Information" on page 1
"Chapter 2. Activity Log for Problem Determination" on page 3

# Appendix A. Activity Log Samples

This appendix contains the activity log samples for “Reading the Activity Log” on page 6. They include:



## Activity Log Sample 1

This is the sample activity log for “Activity Log Sample 1: Server Does Not Start” on page 8. Some text has been wrapped.

```
ComponentId: 102
ProcessId: 355
ThreadId: 404
FunctionName: SOM_CreateClass
ProbeId: 411
SourceId: 1.102 src/shasta/somrt/somrt.C
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.518456938
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOM_CreateClass:411 reported an activity.
ExtendedMessage: Class IQueryLocalObjectImpl::ParameterListBuilder already exists
RawDataLen: 0
-------------------------------------

ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::run()
ProbeId: 2417
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.573342110
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::run():2417 reported an activity.
ExtendedMessage: Server fred Name Server is starting
RawDataLen: 0
-------------------------------------
```

```
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::contextControlInit()
ProbeId: 2778
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.708997023
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::contextControlInit():2778 reported an
  activity.
ExtendedMessage: Start ContextControl DLL Initialization
RawDataLen: 0
-------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1054
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.717757631
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1054 reported an
  activity.
ExtendedMessage: Start DLL: somsmsri.dll Function: SOMInit Timeout: 120000
RawDataLen: 0
-------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1245
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.794630229
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1245 reported an
  activity.
ExtendedMessage: End DLL: somsmsri.dll Function: SOMInit
```

```
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1054
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.803796475
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1054 reported an
  activity.
ExtendedMessage: Start DLL: somrssri.dll Function: SOMInit Timeout: 60000
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1245
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.839203479
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1245 reported an
  activity.
ExtendedMessage: End DLL: somrssri.dll Function: SOMInit
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1054
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.848279211
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1054 reported an
```

```
  activity.
ExtendedMessage: Start DLL: somscs1i.dll Function: sec_ctx_control_init Timeout: 60000
RawDataLen: 0
-------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: startSeqDll(const SOMString&,void*)
ProbeId: 1245
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.964275765
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function startSeqDll(const SOMString&,void*):1245 reported an
  activity.
ExtendedMessage: End DLL: somscs1i.dll Function: sec_ctx_control_init
RawDataLen: 0
-------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::contextControlInit()
ProbeId: 2795
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.977575496
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::contextControlInit():2795 reported an
  activity.
ExtendedMessage: End ContextControl DLL Initialization
RawDataLen: 0
-------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::initIM()
ProbeId: 2884
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:34.980611915
UnitOfWork:
Severity: 3
Category: 2
```

```
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::initIM():2884 reported an activity.
ExtendedMessage: Start Instance Manager Initialization #1
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 355
ThreadId: 404
FunctionName: IBOIMSystemObject_IContainerImpl::processState()
ProbeId: 976
SourceId: 1.73 src/instancemgr/boim/server/IBOIMSystemObject_IContainerImpl_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:35.066616397
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function IBOIMSystemObject_IContainerImpl::processState():976
  reported an activity.
ExtendedMessage: The state attribute of container 'BOIM Master Container' is changed.
  The new value is 3. Value 1=absent, 2=exists, 3=created, 4=initialized, 5=changed,
  6=dead,7=awaitingDisposal, 8=partiallyChanged, and 9=partiallyDestroyed. Other
  values are possible.
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::initIM()
ProbeId: 2903
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:35.072504015
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::initIM():2903 reported an
  activity.
ExtendedMessage: End Instance Manager Initialization #1
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::initORB()
ProbeId: 2982
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
```

```
TimeStamp: 9/28/98 15:59:35.075451596
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::initORB():2982 reported an
  activity.
ExtendedMessage: Start ORB Initialization
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:36.830530871
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:36.833896661
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
```

```
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:38.332923633
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:38.336298642
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:39.834734756
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
```

```
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:39.837931251
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:41.336940623
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:41.340454755
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
```

```
FunctionName:
CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:41.344517002
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags) line 829.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:43.339797689
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:43.343165993
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
```

```
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:44.841931479
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:44.845462374
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:46.344211098
UnitOfWork:
```

```
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:46.349173459
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:47.846250183
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
```

```
clientUserId:
TimeStamp: 9/28/98 15:59:47.849612621
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:47.853570106
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags) line 829.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:49.849060316
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
```

```
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:49.852330563
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------

ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:51.351251097
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------

ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:51.354459325
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
```

```
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:52.853443553
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:52.856646753
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:54.355628467
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
```

```
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:54.358821610
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:54.362717076
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags) line 829.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
```

```
clientUserId:
TimeStamp: 9/28/98 15:59:56.358513191
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:56.361708847
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:57.860646981
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
```

```
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:57.863848504
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:59.362812619
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:59:59.366001570
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
```

```
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:00.864984961
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:00.868238446
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:00.872165760
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
```

```
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags) line 829.
RawDataLen: 0
-----------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:02.867856275
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-----------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:02.871060313
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
 (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-----------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:04.370048732
UnitOfWork:
Severity: 1
Category: 2
```

```
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:04.373413684
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:05.872174141
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900
  returned error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:05.875327055
```

```
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:07.374361569
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:07.377531245
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
```

```
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:07.381414140
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
  line 829.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:09.377245454
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:09.380522406
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
```

```
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:10.879364159
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:10.882600882
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-----------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:12.381562482
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
```

```
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:12.384741377
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.883664425
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage: Operating System call 'connect' : 9.5.73.52, 900 returned
  error code 10061.
RawDataLen: 0
-------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName: transip.cpp
ProbeId: 1067
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.886824882
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
```

```
PrimaryMessage: The function transip.cpp:1067 reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at transip.cpp line 1067.
RawDataLen: 0
------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
 CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags)
ProbeId: 829
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.892388995
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags):829
  reported an activity.
ExtendedMessage:
  A SystemException occurred: COMM_FAILURE, minor code 1229062206
  (SOMDERROR_CannotConnect) at
  CORBA::Request::Request(CORBA::Object_ORBProxy_ptr,char*,CORBA::Flags) line 829.
RawDataLen: 0
------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
FunctionName:
  CallStreamMgr::start_all_listening(::ImplementationDef&,Waitpoint&,Queue&,
  unsigned char)
ProbeId: 886
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.896327204
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CallStreamMgr::start_all_listening(::ImplementationDef&,Waitpoint&,Queue&,unsigned
  char):886 reported an activity.
ExtendedMessage:
  A SystemException occurred: INITIALIZE, minor code 1229062222
  (SOMDERROR_SOMDDNotRunning) at
  CallStreamMgr::start_all_listening(::ImplementationDef&,Waitpoint&,Queue&,unsigned char)
  line 886.
RawDataLen: 0
------------------------------------
ComponentId: 393316
ProcessId: 355
ThreadId: 404
```

```
FunctionName:
CORBA::BOA::initialize_impl(CORBA::ImplementationDef*,CORBA::Boolean)
ProbeId: 1956
SourceId: 1.20 src/orb/src/somd/somderr.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.899942746
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  CORBA::BOA::initialize_impl(CORBA::ImplementationDef*,CORBA::Boolean):1956
  reported an activity.
ExtendedMessage:
  A SystemException occurred: INITIALIZE, minor code 1229062222
  (SOMDERROR_SOMDDNotRunning) at
  CORBA::BOA::initialize_impl(CORBA::ImplementationDef*,CORBA::Boolean) line 1956.
RawDataLen: 0
-----------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::initORB()
ProbeId: 3049
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.932630132
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::initORB():3049 reported an activity.
ExtendedMessage: Location Service Daemon is not started
RawDataLen: 0
-----------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: main(int,char**)
ProbeId: 98
SourceId: 1.13 src/sr/somsrsm.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:13.935781369
UnitOfWork:
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function main(int,char**):98 reported an activity.
ExtendedMessage: Server fred Name Server failed
```

```
RawDataLen: 0
------------------------------------------
ComponentId: 103
ProcessId: 355
ThreadId: 404
FunctionName: SOMSRSMClass::~SOMSRSMClass()
ProbeId: 2358
SourceId: 1.63.1.7 src/sr/somsrsmc.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: fred Name Server
clientHostName:
clientUserId:
TimeStamp: 9/28/98 16:00:14.004066845
UnitOfWork:
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function SOMSRSMClass::~SOMSRSMClass():2358 reported an
  activity.
ExtendedMessage: Server fred Name Server is stopped
RawDataLen: 0
73 records found and printed.
------------------------------------------
```

**RELATED REFERENCES**

"Reading the Activity Log" on page 6

## Activity Log Sample 2

This is the sample activity log for "Activity Log Sample 2: Client Application Receives an Expected Error" on page 8. Some text has been wrapped.

```
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMExtLocalToServer_IDataObjectBase_Impl::retrieveFromDataStore()
ProbeId: 415 SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtLocalToServer_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.177520520
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtLocalToServer_IDataObjectBase_Impl::retrieveFromDataStore():415
  reraised CORBA exception IBOIMException::IDataKeyNotFound.
ExtendedMessage:
RawDataLen: 0
------------------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMLocalToServer_IMMixinBase::internalMixinRetrieveFromDatastore()
ProbeId: 3892
SourceId: 1.105 src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
```

```
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.203674116
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinBase::internalMixinRetrieveFromDatastore():3892
  received CORBA exception IBOIMException::IDataKeyNotFound and raised CORBA
  exception IManagedClient::INoObjectWKey.
ExtendedMessage: The Data Object 'retrieveFromDataStore()' method raised a
  CORBA exception, IBOIMException::IDataKeyNotFound. The exception is mapped
  to IManagedClient::INoObjectWKey.
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMLocalToServer_IMMixinBase::internalMixinRefreshFromDatastore()
ProbeId: 2820
SourceId: 1.105 src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.207259487
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinBase::internalMixinRefreshFromDatastore():2820
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMLocalToServer_IMMixinBase::restorePersistentData()
ProbeId: 2042
SourceId: 1.105 src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.210035258
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinBase::restorePersistentData():2042
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
```

```
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData()
ProbeId: 1129
SourceId: 1.66
  src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.212795106
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData():1129
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName:
  IBOIMLocalToServer_IMMixinTransactionalBase::synchronizeWithBackingStore()
ProbeId: 668
SourceId: 1.66
  src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.215638762
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinTransactionalBase::synchronizeWithBackingStore():668
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
  ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
  IBOIMExtLocalToServer::IDataObject*,CORBA::Boolean,CORBA::Boolean,
  IHomeTransactionHelper*)
ProbeId: 7819
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
```

```
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.218501695
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
  ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
  IBOIMExtLocalToServer::IDataObject*,CORBA::Boolean,CORBA::Boolean,
  IHomeTransactionHelper*):7819
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
  (IManagedLocal::IPrimaryKey_ptr,const ByteString&)
ProbeId: 5835
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.223648437
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
  (IManagedLocal::IPrimaryKey_ptr,const ByteString&):5835
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName: IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5643
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.226974837
UnitOfWork: 12455:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&):5643
  reported an activity.
ExtendedMessage: The Home 'Home for making Policy objects.' reported an error:
  The Home could not find the Managed Object.
  The Managed Object creation function was: 'PolicyEmSQLMO_create'.
  The managedObjectDllName was 'PolicyDB2.dll'. The
```

```
  dataObjectCreateFunctionName was 'PolicyEmSQLDOImpl_create',
  the dataObjectDllName was 'PolicyDB2.dll'. The Primary
  keyCreateFunctionName was: 'PolicyKey_create'. The Primary
  keyDllName was 'PolicyC.dll'. Its Key was:
RawDataLen: 12
RawData:
0000 02 01 00 01 52 03 00 00 - A4 09 00 00 ....R.......
------------------------------------
ComponentId: 131175
ProcessId: 475
ThreadId: 366
FunctionName:
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5649
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:05.230031370
UnitOfWork: 12455:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&):5649
  reraised CORBA exception IManagedClient::INoObjectWKey.
ExtendedMessage:
RawDataLen: 0
------------------------------------
```

**RELATED REFERENCES**

"Reading the Activity Log" on page 6

## Activity Log Sample 3

This is the sample activity log for "Activity Log Sample 3: Client Application Receives an Error And Server Dies" on page 9. Some text has been wrapped.

```
------------------------------------
ComponentId: 104
ProcessId: 475
ThreadId: 366
FunctionName: handleSignal(int,int,CONTEXT*)
ProbeId: 608
SourceId: 1.29 src/ras/src/raswinex.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:12.701712974
UnitOfWork: 7319:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function handleSignal(int,int,CONTEXT*):608 reported an error.
ExtendedMessage: A system error was detected - 40 SEG-VIOLATION (OS signal nbr: 2).
Intel context flags:
```

```
i386/486 CONTROL INTEGER SEGMENT FULL FLOAT-POINT DEBUG-REG
Intel segment registers:
Gs:00000000 Fs:0000003b Es:00000023 Ds:00000023 Ss:00000023 Cs:0000001b
Intel GP registers:
Edi:05d74740 Esi:022559af Ebx:05d74740 Edx:000009a4 Ecx:00000000
Eax:00000000 Ebp:02ebf42c Eip:05a261fa Esp:02ebf3fc
Error occurred at (Eip):05a261fa, in source file:PolicyS.dll
Call stack unwind: base pointer (return address - DLL/EXE name)
02ebf42c (05819120 - PolicyDB2.dll)
02ebf48c (058E9054 - PolicyC.dll)
02ebf4c4 (058E998A - PolicyC.dll)
2ebf584 (059D5190 - PolicyS.dll)
02ebf5d0 (6B6859E7 - somorori.dll)
02ebf828 (6B6857F8 - somorori.dll)
02ebfc0c (6EACDCE5 - somsrsai.dll)
02ebfd80 (6EACFB5D - somsrsai.dll)
02ebff54 (504419F7 - cppobi36.dll)
02ebffa4 (5043F663 - cppobi36.dll)
02ebffb8 (77F04F2C - KERNEL32.dll)
Module list: size (load address) date-linked-GMT => DLL/EXE name
512000 (00240000) 1998/04/18 04:27:22 GMT => c:\IBMCPPW\BIN\CPPWM35I.dll
19456 (00360000) 1998/09/09 10:46:19 GMT => e:\e9836.01.lite\lib.nt\teceifi.dll
289280 (00400000) 1998/09/09 11:22:59 GMT => e:\e9836.01.lite\bin.nt\somsrsm.exe
4605952 (004C0000) 1998/09/22 19:17:03 GMT => e:\e9836.01.lite\lib.nt\somibe1i.dll
5027328 (009B0000) 1998/09/16 18:47:26 GMT => e:\e9836.01.lite\lib.nt\somibs1i.dll
57856 (02490000) 1998/04/18 04:17:07 GMT => C:\IBMCPPW\LOCALE\iconv\UCSTBL.dll
16896 (030C0000) 1998/09/09 09:06:58 GMT =>
e:\e9836.01.lite\lib.nt\EN_US\somscl1i.dll
980992 (03350000) 1997/08/16 12:53:48 GMT => C:\SQLLIB\BIN\DB2APP.dll
1257472 (03450000) 1997/08/16 12:53:43 GMT => C:\SQLLIB\BIN\DB2SYS.dll
14848 (03590000) 1997/08/16 12:53:51 GMT => C:\SQLLIB\BIN\DB2WINT.dll
32768 (035A0000) 1997/08/16 12:53:44 GMT => C:\SQLLIB\BIN\DB2SYSP.dll
102400 (035B0000) 1997/08/16 12:53:37 GMT => C:\SQLLIB\BIN\DB2ABIND.dll
17920 (035D0000) 1997/08/16 12:55:59 GMT => C:\SQLLIB\BIN\DB2CAPRO.dll
37648 (03A10000) 1997/04/22 23:50:00 GMT => C:\WINNT\System32\rnr20.dll
67072 (04A00000) 1997/04/25 23:13:52 GMT => C:\WINNT\System32\odbcint.dll
230672 (04A50000) 1997/04/25 23:13:44 GMT => C:\WINNT\System32\ODBC32.dll
4096 (04BA0000) 1998/09/09 10:09:57 GMT => e:\e9836.01.lite\lib.nt\db2slf.dll
893952 (057A0000) 1998/09/09 11:08:23 GMT => e:\e9836.01.lite\lib.nt\PolicyDB2.dll
326656 (058D0000) 1998/09/09 10:54:30 GMT => e:\e9836.01.lite\lib.nt\PolicyC.dll
2835456 (05980000) 1998/09/28 15:20:32 GMT => e:\e9836.01.lite\lib.nt\PolicyS.dll
1972224 (05F00000) 1998/09/09 11:21:11 GMT => e:\e9836.01.lite\lib.nt\JPolicyS.dll
414208 (06160000) 1998/04/16 11:05:48 GMT =>
e:\e9836.01.lite\tools\nt\java\bin\javai.dll
46080 (08670000) 1998/04/16 11:39:15 GMT =>
e:\e9836.01.lite\tools\nt\java\bin\zip.dll
3240256 (10000000) 1998/01/16 22:10:53 GMT => C:\DCE\dcelocal\bin\libdce.dll
193024 (1C000000) 1998/01/12 19:34:43 GMT => C:\DCE\dcelocal\bin\pthreads.dll
162816 (28100000) 1997/09/15 15:58:32 GMT => e:\e9836.01.lite\lib.nt\setloc1.dll
519680 (50000000) 1997/09/15 18:00:13 GMT => c:\IBMCPPW\BIN\CPPWOB3I.dll
1283072 (50400000) 1997/11/05 22:05:46 GMT =>
e:\e9836.01.lite\tools\nt\ioc.t\bin\cppobi36.dll
2023424 (51000000) 1997/09/15 18:01:28 GMT => c:\IBMCPPW\BIN\CPPWOU3I.dll
13584 (5F810000) 1997/04/28 21:57:58 GMT => C:\WINNT\System32\rpcltc1.dll
313344 (68000000) 1998/09/09 13:02:41 GMT => e:\e9836.01.lite\lib.nt\somcscdi.dll
188416 (680E0000) 1998/09/09 13:02:41 GMT => e:\e9836.01.lite\lib.nt\somcscsi.dll
7168 (68170000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsdli.dll
32768 (681F0000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsmmi.dll
104960 (683C0000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcssni.dll
19968 (68460000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcssti.dll
6656 (68500000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsthi.dll
14336 (68580000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsuii.dll
794112 (68620000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somdc01i.dll
127488 (68760000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somdc02i.dll
396800 (688D0000) 1998/09/09 13:02:43 GMT => e:\e9836.01.lite\lib.nt\somdt00i.dll
24064 (68CD0000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somi2a1i.dll
289280 (68D60000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somi2s1i.dll
```

```
3622400 (68E10000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somiba1i.dll
210944 (691F0000) 1998/09/09 13:02:47 GMT => e:\e9836.01.lite\lib.nt\somibb1i.dll
855552 (69410000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibi1i.dll
166400 (69530000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibl1i.dll
373760 (695C0000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibp1i.dll
175104 (69900000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somida1i.dll
58368 (699A0000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somide1i.dll
256512 (69A30000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somids1i.dll
659456 (69AF0000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somima1i.dll
50176 (69BF0000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somimb1i.dll
73728 (69C80000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somims1i.dll
510976 (6A1A0000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somira1i.dll
195584 (6A280000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somirs1i.dll
57344 (6A310000) 1998/09/09 13:02:51 GMT => e:\e9836.01.lite\lib.nt\somiym1i.dll
2354176 (6A3A0000) 1998/09/09 13:02:51 GMT => e:\e9836.01.lite\lib.nt\somlcm1i.dll
790528 (6A650000) 1998/09/09 13:02:52 GMT => e:\e9836.01.lite\lib.nt\somloc1i.dll
3214336 (6AF70000) 1998/09/09 13:02:56 GMT => e:\e9836.01.lite\lib.nt\somoq01i.dll
45568 (6B310000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somorbti.dll
182272 (6B3A0000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somorcdi.dll
693248 (6B4D0000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somoriri.dll
1086976 (6B660000) 1998/09/09 13:02:59 GMT => e:\e9836.01.lite\lib.nt\somorori.dll
173568 (6B9E0000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somorpti.dll
24064 (6BA90000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somorssi.dll
6035456 (6BB30000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somosb1i.dll
2080768 (6C150000) 1998/09/09 13:03:01 GMT => e:\e9836.01.lite\lib.nt\somosc1i.dll
1216512 (6C3C0000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmcii.dll
9728 (6C550000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmg1i.dll
1097216 (6C5F0000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmsii.dll
393216 (6CF10000) 1998/09/09 13:03:05 GMT => e:\e9836.01.lite\lib.nt\somrsbsi.dll
25088 (6D0F0000) 1998/09/09 13:03:06 GMT => e:\e9836.01.lite\lib.nt\somrsori.dll
125440 (6D180000) 1998/09/09 13:03:06 GMT => e:\e9836.01.lite\lib.nt\somrssri.dll
1385984 (6D440000) 1998/09/09 13:03:07 GMT => e:\e9836.01.lite\lib.nt\somscs1i.dll
638976 (6D600000) 1998/09/09 13:03:08 GMT => e:\e9836.01.lite\lib.nt\somsgcli.dll
709120 (6DAA0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somsgtsi.dll
48640 (6DBD0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somsh.dll
115200 (6DCC0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somshcb.dll
99840 (6DD70000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somshcpi.dll
91648 (6DE00000) 1998/09/09 13:03:10 GMT => e:\e9836.01.lite\lib.nt\somshori.dll
20992 (6E290000) 1998/09/09 13:03:10 GMT => e:\e9836.01.lite\lib.nt\somsmdei.dll
160768 (6E4B0000) 1998/09/09 13:03:11 GMT => e:\e9836.01.lite\lib.nt\somsmfni.dll
101376 (6E790000) 1998/09/09 13:03:11 GMT => e:\e9836.01.lite\lib.nt\somsmoii.dll
52736 (6EA20000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsmsri.dll
965632 (6EA90000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsrsai.dll
671744 (6EBE0000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsrssi.dll
2754560 (6EDA0000) 1998/09/09 13:03:13 GMT => e:\e9836.01.lite\lib.nt\somsss1i.dll
516096 (6F2F0000) 1998/09/09 13:03:17 GMT => e:\e9836.01.lite\lib.nt\somtrm1i.dll
2202112 (6F3F0000) 1998/09/09 13:03:17 GMT => e:\e9836.01.lite\lib.nt\somtrs1i.dll
2886144 (6F670000) 1998/09/09 13:03:18 GMT => e:\e9836.01.lite\lib.nt\somtrt1i.dll
427520 (6F9A0000) 1998/09/09 13:03:20 GMT => e:\e9836.01.lite\lib.nt\somtrx1i.dll
297984 (74C00000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\skit.dll
484352 (74C60000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\x509cms.dll
62976 (74CF0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\soedber.dll
34816 (74D10000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\soedapi.dll
13312 (74D50000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\ossdmem.dll
12288 (74D60000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\ossapi.dll
28160 (74D70000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\nspcommon.dll
18432 (74E00000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cstrain.dll
5120 (74E30000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cmskfra.dll
1566208 (74E50000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cms.dll
12288 (74FE0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cmpval.dll
13824 (74FF0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\berreal.dll
44304 (77660000) 1997/04/17 22:34:19 GMT => C:\WINNT\system32\msafd.dll
18704 (77690000) 1997/03/28 01:58:49 GMT => C:\WINNT\System32\wshtcpip.dll
8464 (776A0000) 1997/01/02 19:54:11 GMT => C:\WINNT\System32\WS2HELP.dll
59664 (776B0000) 1997/04/03 01:10:00 GMT => C:\WINNT\System32\WS2_32.dll
20240 (776D0000) 1996/07/26 19:19:24 GMT => C:\WINNT\System32\WSOCK32.dll
41744 (777E0000) 1997/04/22 23:50:01 GMT => C:\WINNT\System32\SAMLIB.dll
```

```
224528 (77800000) 1997/04/25 19:33:39 GMT => C:\WINNT\System32\NETAPI32.dll
17168 (77840000) 1996/07/26 19:19:23 GMT => C:\WINNT\System32\NETRAP.dll
12560 (779C0000) 1996/07/26 19:19:21 GMT => C:\WINNT\system32\LZ32.dll
65024 (779D0000) 1996/07/26 19:19:21 GMT => C:\WINNT\System32\MSVCRT40.dll
36112 (77A90000) 1996/07/26 19:19:20 GMT => C:\WINNT\system32\VERSION.dll
704272 (77B20000) 1997/04/25 19:33:41 GMT => C:\WINNT\system32\ole32.dll
310032 (77BF0000) 1997/04/25 19:33:18 GMT => C:\WINNT\system32\COMCTL32.dll
1277200 (77C40000) 1997/04/25 19:33:49 GMT => C:\WINNT\system32\SHELL32.dll
185104 (77D80000) 1997/04/25 19:33:18 GMT => C:\WINNT\system32\comdlg32.dll
246544 (77DC0000) 1997/04/11 23:24:11 GMT => C:\WINNT\system32\ADVAPI32.dll
317200 (77E10000) 1997/04/28 21:57:58 GMT => C:\WINNT\system32\RPCRT4.dll
330512 (77E70000) 1997/04/25 19:33:52 GMT => C:\WINNT\system32\USER32.dll
165648 (77ED0000) 1997/04/25 19:33:25 GMT => C:\WINNT\system32\GDI32.dll
372496 (77F00000) 1997/04/25 19:33:31 GMT => C:\WINNT\system32\KERNEL32.dll
355088 (77F60000) 1997/04/11 20:38:50 GMT => C:\WINNT\System32\ntdll.dll
149264 (77FD0000) 1996/07/17 18:15:07 GMT => C:\WINNT\System32\WINMM.dll
271632 (78000000) 1997/01/23 07:07:13 GMT => C:\WINNT\system32\MSVCRT.dll
70656 (780A0000) 1997/01/23 05:27:47 GMT => C:\WINNT\System32\MSVCIRT.dll
Storage dump:
02EBF40C AA AA AA AA 40 AD AD 05 - AA AA AA AA AA AA AA AA ....@...........
02EBF41C 00 00 00 00 DC FF FF FF - 84 F4 EB 02 90 D0 A8 05 ................
02EBF42C 8C F4 EB 02 20 91 81 05 - 0C 4C D7 05 AF 59 25 02 .... ....L...Y%.
02EBF43C 3B 01 92 05 F0 34 87 05 - 02 00 00 00 38 F4 EB 02 ;....4......8...
02EBF44C DC BA 86 05 50 9D EE 05 - 30 42 85 05 00 00 00 00 ....P...0B......
02EBF45C E0 FF FF FF D0 B5 D1 05 - 0C 4C D7 05 6E 01 00 00 .........L..n...
02EBF46C A7 D7 F3 6C D0 B5 D1 05 - 01 9F EE 05 F0 64 D7 05 ...l.........d..
02EBF47C D0 4A D7 05 B8 FF FF FF - BC F4 EB 02 90 D0 82 05 .J..............
02EBF48C C4 F4 EB 02 54 90 8E 05 - 94 86 EE 05 3B 01 92 05 ....T.......;...
02EBF49C 01 00 00 00 A4 10 92 05 - E4 15 92 05 50 A8 F1 6E ............P..n
02EBF4AC 84 F5 EB 02 9C C9 67 6B - B0 48 D7 05 D8 FF FF FF ......gk.H......
02EBF4BC 7C F5 EB 02 40 B1 90 05 - 84 F5 EB 02 8A 99 8E 05 |...@...........
02EBF4CC B0 9C EE 05 94 86 EE 05 - F8 86 EE 05 30 68 25 02 ............0h%.
02EBF4DC 0C 00 00 00 D0 F4 EB 02 - 84 0F 92 05 00 00 FD 00 ................
02EBF4EC 20 00 00 00 E0 30 3F 00 - 00 00 00 00 00 00 FD 00 ....0?.........
02EBF4FC F8 9C EE 05 48 F5 EB 02 - 49 1C 25 00 F0 30 3F 00 ....H...I.%..0?.
02EBF50C 00 6B 79 05 78 66 79 05 - 20 F8 EB 02 F0 30 3F 00 .ky.xfy. ....0?.
02EBF51C 6B 16 41 50 40 00 00 00 - 34 72 52 50 00 00 00 00 k.AP@...4rRP....
-- END OF EXCEPTION DATA --
RawDataLen: 0
------------------------------------
ComponentId: 103
ProcessId: 475
ThreadId: 366
FunctionName: SOMSR_ContinueDispatchCallbackObject::wrapExecute(void*)
ProbeId: 419
SourceId: 1.20.1.3 src/sr/somsrcb.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:12.725450342
UnitOfWork: 7319:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  SOMSR_ContinueDispatchCallbackObject::wrapExecute(void*):419 raised CORBA exception
  CORBA::UNKNOWN, error code is 0x4942009C SRThreadManagerUnknownError.
ExtendedMessage: An IException object was received. The exception name is
  SOMRASOperatingSystemException, the errorId is 40, and recoverable is 0.
  The exception text stack follows:
  40 SEG-VIOLATION
  The exception location stack follows:
RawDataLen: 0
```

```
------------------------------------
ComponentId: 103
ProcessId: 475
ThreadId: 366
FunctionName:
SOMSR_ContinueDispatchCallbackObject::dumpTargetInfo(void*)
ProbeId: 301
SourceId: 1.20.1.3 src/sr/somsrcb.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 14:46:12.729104436
UnitOfWork: 7319:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  SOMSR_ContinueDispatchCallbackObject::dumpTargetInfo(void*):301 reported an
  activity.
ExtendedMessage: SRThreadManagerUnknownError: additional information ->
  —>Method Name: addBeneficiary
  —>Target classname: PolicyEmSQLMO
  —>Target type_id: IDL:PolicyEmSQLMO:1.0
  —>Target refcount: 1
RawDataLen: 0
------------------------------------
```

**RELATED REFERENCES**

"Reading the Activity Log" on page 6

## Activity Log Sample 4

This is the sample activity log for "Activity Log Sample 4: Client Application Receives an Error" on page 10. Some text has been wrapped.

```
ComponentId: 262253
ProcessId: 179
ThreadId: 216
FunctionName: OTSXAConnection::traceXAResult(char*,int,OTSXAXId*,int)
ProbeId: 1778
SourceId: 1.7 src/objsvcs/transactions/xa/otsxacnn.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.192994192
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  OTSXAConnection::traceXAResult(char*,int,OTSXAXId*,int):1778 reported an
  activity.
ExtendedMessage: Attempt in server testsrv to open a
  connection to the XA Resource Manager database POLICY2 failed with return
  code XAER_RMERR.
RawDataLen: 0
```

```
------------------------------------------------
ComponentId: 262253
ProcessId: 179
ThreadId: 216
FunctionName:
 OTSXAConnection::OTSXAConnection(OTSXADatabase*,CORBA::ULong)
ProbeId: 212
SourceId: 1.7 src/objsvcs/transactions/xa/otsxacnn.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.196270306
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  OTSXAConnection::OTSXAConnection(OTSXADatabase*,CORBA::ULong):212
  raised CORBA exception CORBA::INTERNAL, error code is 0x494202F0
  SOMTRRAS::Minor_unexpectedRetCode.
ExtendedMessage:
RawDataLen: 0
------------------------------------------------
ComponentId: 262253
ProcessId: 179
ThreadId: 216
FunctionName: OTSXAContext::connectdB(CORBA::ULong)
ProbeId: 247
SourceId: 1.3 src/objsvcs/transactions/xa/otsxactx.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.227013311
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function OTSXAContext::connectdB(CORBA::ULong):247
  raised CORBA exception CORBA::INTERNAL, error code is 0x494202EF
  SOMTRRAS::Minor_unexpectedException.
ExtendedMessage:
RawDataLen: 0
------------------------------------------------
ComponentId: 131178
ProcessId: 179
ThreadId: 216
FunctionName: IDB2AALocalToServer_DB2Context::connectdB()
ProbeId: 335
SourceId: 1.5
 src/instancemgr/db2aa/server/IDB2AALocalToServer_DB2ContextImpl_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.267224276
UnitOfWork: 25214:fed
```

```
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IDB2AALocalToServer_DB2Context::connectdB():335 reported an activity.
ExtendedMessage: connect method failed when invoked on the OTSXAContext.
RawDataLen: 0
-------------------------------------
ComponentId: 131178
ProcessId: 179
ThreadId: 216
FunctionName:
 IDB2AALocalToServer_DB2ContextControl::connect(CORBA::ULong)
ProbeId: 927
SourceId: 1.11
 src/instancemgr/db2aa/server/IDB2AALocalToServer_DB2ContextControlImpl_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.281208731
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IDB2AALocalToServer_DB2ContextControl::connect(CORBA::ULong):927
  reported an activity.
ExtendedMessage: beginXA or connectdB method failed when invoked on the
  IDB2AALocalToServer_DB2Context.
RawDataLen: 0
-------------------------------------
ComponentId: 131178
ProcessId: 179
ThreadId: 216
FunctionName:
 IDB2AALocalToServer_DB2ContextControl::connect(CORBA::ULong)
ProbeId: 988
SourceId: 1.11
 src/instancemgr/db2aa/server/IDB2AALocalToServer_DB2ContextControlImpl_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.284357455
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IDB2AALocalToServer_DB2ContextControl::connect(CORBA::ULong):988 reported
  an activity.
ExtendedMessage: connect method failed when invoked on the
  IDB2AALocalToServer_DB2ContextControl.
RawDataLen: 0
-------------------------------------
ComponentId: 131178
ProcessId: 179
ThreadId: 216
FunctionName: IDB2AALocalToServer_IDB2ConnectionMgr_Impl::connect(CORBA::ULong)
ProbeId: 55
```

```
SourceId: 1.1
 src/instancemgr/db2aa/server/IDB2AALocalToServer_DB2ConnectionMgrImpl_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.293382901
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IDB2AALocalToServer_IDB2ConnectionMgr_Impl::connect(CORBA::ULong):55
  raised CORBA exception IRDBIMLocalToServer::IXAConnectionFailed, error
  code is 0x0 0.
ExtendedMessage: connect method failed when invoked on the
  IDB2AALocalToServer_IDB2ConnectionMgr_Impl.
RawDataLen: 0
-------------------------------------
ComponentId: 131176
ProcessId: 179
ThreadId: 216
FunctionName: IRDBIMExtLocalToServer_IDataObject_Impl::connect()
ProbeId: 367
SourceId: 1.3
 src/instancemgr/rdbim/extendable/IRDBIMExtLocalToServer_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.296704272
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
IRDBIMExtLocalToServer_IDataObject_Impl::connect():367 reraised CORBA exception
IRDBIMLocalToServer::IXAConnectionFailed.
ExtendedMessage:
RawDataLen: 0
-------------------------------------
ComponentId: 131176
ProcessId: 179
ThreadId: 216
FunctionName: IRDBIMExtLocalToServer_IDataObject_Impl::retrieveFromDataStore()
ProbeId: 215
SourceId: 1.3
 src/instancemgr/rdbim/extendable/IRDBIMExtLocalToServer_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.299620843
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
```

```
  IRDBIMExtLocalToServer_IDataObject_Impl::retrieveFromDataStore():215
  reraised CORBA exception IRDBIMLocalToServer::IXAConnectionFailed.
ExtendedMessage:
RawDataLen: 0
-------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
 IBOIMLocalToServer_IMMixinBase::internalMixinRetrieveFromDatastore()
ProbeId: 3898
SourceId: 1.105
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.326380381
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinBase::internalMixinRetrieveFromDatastore():3898
  received CORBA exception IRDBIMLocalToServer::IXAConnectionFailed and raised
  CORBA exception CORBA::UNKNOWN.
ExtendedMessage: The Data Object 'retrieveFromDataStore()' method raised
  a CORBA exception, IRDBIMLocalToServer::IXAConnectionFailed. The exception is
  mapped to CORBA::UNKNOWN.
RawDataLen: 0
-------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
 IBOIMLocalToServer_IMMixinBase::internalMixinRefreshFromDatastore()
ProbeId: 2820
SourceId: 1.105
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.330106552
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
IBOIMLocalToServer_IMMixinBase::internalMixinRefreshFromDatastore():2820
  reraised CORBA exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
-------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName: IBOIMLocalToServer_IMMixinBase::restorePersistentData()
ProbeId: 2042
SourceId: 1.105
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinBase_I.cpp
```

```
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.333009714
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinBase::restorePersistentData():2042 reraised CORBA
  exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
-------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
  IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData()
ProbeId: 1133
SourceId: 1.66
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.335789675
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData():1133
  reraised CORBA exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
-------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
  IBOIMLocalToServer_IMMixinTransactionalBase::synchronizeWithBackingStore()
ProbeId: 668
SourceId: 1.66
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.339541827
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMLocalToServer_IMMixinTransactionalBase::synchronizeWithBackingStore():668
  reraised CORBA exception CORBA::UNKNOWN.
```

```
ExtendedMessage:
RawDataLen: 0
------------------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
 IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
 ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
 IBOIMExtLocalToServer::IDataObject*,
 CORBA::Boolean,CORBA::Boolean,IHomeTransactionHelper*)
ProbeId: 7847
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.342558970
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
  ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
  IBOIMExtLocalToServer::IDataObject*,CORBA::Boolean,CORBA::Boolean,
  IHomeTransactionHelper*):7847 reraised CORBA exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
------------------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
 IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
 (IManagedLocal::IPrimaryKey_ptr,const ByteString&)
ProbeId: 5849
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.348621750
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
  (IManagedLocal::IPrimaryKey_ptr,const ByteString&):5849
  reraised CORBA exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
------------------------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5643
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
```

```
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.351970778
UnitOfWork: 25214:fed
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&):5643
  reported an activity.
ExtendedMessage: The Home 'Home for making Policy objects.' reported an
  error: The Home could not find the Managed Object. The Managed Object creation
  function was: 'PolicyEmSQLMO_create'. The managedObjectDllName was
  'PolicyDB2.dll'. The dataObjectCreateFunctionName was 'PolicyEmSQLDOImpl_create',
  the dataObjectDllName was 'PolicyDB2.dll'. The Primary keyCreateFunctionName was:
  'PolicyKey_create'. The Primary keyDllName was 'PolicyC.dll'. Its Key was:
RawDataLen: 12
RawData:
0000 02 01 00 01 52 03 00 00 - A4 09 00 00 ....R.......
-------------------------------
ComponentId: 131175
ProcessId: 179
ThreadId: 216
FunctionName:
IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5662
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/29/98 12:48:02.355266168
UnitOfWork: 25214:fed
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
  IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&):5662
  reraised CORBA exception CORBA::UNKNOWN.
ExtendedMessage:
RawDataLen: 0
102 records found and printed.
-------------------------------
```

**RELATED REFERENCES**

"Reading the Activity Log" on page 6

## Activity Log Sample 5

This is the sample activity log for "Activity Log Sample 5: Client Application Receives an Error" on page 10. Some text has been wrapped.

```
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
```

```
 IBOIMLocalToServer_IMMixinTransactionalBase::internalMixinGetTransactionCoordinator()
ProbeId: 958
SourceId: 1.66
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.661163269
UnitOfWork: 11475:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMLocalToServer_IMMixinTransactionalBase
 ::internalMixinGetTransactionCoordinator():958
 raised CORBA exception CORBA::TRANSACTION_REQUIRED, error code is 0x0 0.
ExtendedMessage: The before() method processing
 failed when it could not get a Control Transaction Object from the
 Transaction Service. This can mean there was no transactional context
 on the thread.
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
 IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData()
ProbeId: 1031
SourceId: 1.66
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.664751154
UnitOfWork: 11475:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMLocalToServer_IMMixinTransactionalBase::restorePersistentData():1031
 reraised CORBA exception CORBA::TRANSACTION_REQUIRED.
ExtendedMessage:
RawDataLen: 0
-----------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
IBOIMLocalToServer_IMMixinTransactionalBase::synchronizeWithBackingStore()
ProbeId: 668
SourceId: 1.66
src/instancemgr/boim/server/IBOIMLocalToServer_IMMixinTransactionalBase_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
```

```
TimeStamp: 9/28/98 15:27:14.667901553
UnitOfWork: 11475:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMLocalToServer_IMMixinTransactionalBase
 ::synchronizeWithBackingStore():668 reraised
 CORBA exception CORBA::TRANSACTION_REQUIRED.
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
 IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
 ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
 IBOIMExtLocalToServer::IDataObject*,CORBA::Boolean,
 CORBA::Boolean,IHomeTransactionHelper*)
ProbeId: 7847
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.670765324
UnitOfWork: 11475:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMExtSystemObject_IHome_Impl::assembleManagedObject(const
 ByteString*,IBOIMExt::DataObjectState,CORBA::Boolean,CORBA::Boolean,
 IBOIMExtLocalToServer::IDataObject*,CORBA::Boolean,CORBA::Boolean,
 IHomeTransactionHelper*):7847
 reraised CORBA exception CORBA::TRANSACTION_REQUIRED.
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
 IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
 (IManagedLocal::IPrimaryKey_ptr,const ByteString&)
ProbeId: 5849
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.765196015
UnitOfWork: 11475:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMExtSystemObject_IHome_Impl::findGivenPrimaryKey
 (IManagedLocal::IPrimaryKey_ptr,const ByteString&):5849
 reraised CORBA exception CORBA::TRANSACTION_REQUIRED.
```

```
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5643
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.769326148
UnitOfWork: 11475:fred
Severity: 1
Category: 2
FormatWarning: 0
PrimaryMessage: The function
 IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&):5643
 reported an activity.
ExtendedMessage: The Home 'Home for making Policy objects.' reported an
 error: The Home could not find the Managed Object. The Managed Object creation
 function was: 'PolicyEmSQLMO_create'. The managedObjectDllName was 'PolicyDB2.dll'.
 The dataObjectCreateFunctionName was 'PolicyEmSQLDOImpl_create', the
 dataObjectDllName was 'PolicyDB2.dll'. The Primary keyCreateFunctionName was:
 'PolicyKey_create'. The Primary keyDllName was 'PolicyC.dll'. Its Key was:
RawDataLen: 12
RawData:
0000 02 01 00 01 52 03 00 00 - A4 09 00 00 ....R.......
------------------------------------
ComponentId: 131175
ProcessId: 384
ThreadId: 380
FunctionName:
 IBOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString(const ByteString&)
ProbeId: 5662
SourceId: 1.4 src/instancemgr/boim/extendable/IBOIMExtSystemObject_I.cpp
Manufacturer: IBM
Product: Component Broker
Version: 1.3
SOMProcessType: 5
ServerName: testsrv
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:14.773905499
UnitOfWork: 11475:fred
Severity: 3
Category: 2
FormatWarning: 0
PrimaryMessage: The function BOIMExtSystemObject_IHome_Impl::findByPrimaryKeyString
 (const ByteString&):5662 reraised CORBA exception CORBA::TRANSACTION_REQUIRED.
ExtendedMessage:
RawDataLen: 0
------------------------------------
ComponentId: 262253
ProcessId: 408
ThreadId: 472
FunctionName: ICurrent_Impl::rollback()
ProbeId: 1268
SourceId: 1.15 src/objsvcs/transactions/ots/icurrent_I.cpp
Manufacturer: IBM
Product: Component Broker
```

```
Version: 1.3
SOMProcessType: 1
ServerName:
clientHostName:
clientUserId:
TimeStamp: 9/28/98 15:27:15.017074843
UnitOfWork: 1734:fred
Severity: 2
Category: 2
FormatWarning: 0
PrimaryMessage: The function ICurrent_Impl::rollback():1268 raised CORBA
 exception CosTransactions::NoTransaction, error code is 0x494202F9
 SOMTRRAS::Minor_noTransaction.
ExtendedMessage:
RawDataLen: 0
11 records found and printed.
-----------------------------------
```

**RELATED REFERENCES**

"Reading the Activity Log" on page 6

# Appendix B. HandleSignal Log Entry and Map File

This section contains the samples used in “HandleSignal - General Page Fault (GPF) Exception” on page 11. They include:

- “Activity Log Showing General Page Fault Exception”
- “HandleSignal - Compilation Map File Example” on page 164

## Activity Log Showing General Page Fault Exception

The following activity log is used in the discussion of “HandleSignal - General Page Fault (GPF) Exception” on page 11.

```
ComponentId:104
ComponentId:104
ProcessId:387
ThreadId:426
FunctionName:handleSignal(int,int,CONTEXT*)
ProbeId:608
SourceId:1.29 src/ras/src/raswinex.cpp
Manufacturer:IBM
Product:Component Broker
Version:1.3
SOMProcessType: 5
ServerName:testsrv
clientHostName:
clientUserId:
TimeStamp:9/28/98 11:23:35.288653520
UnitOfWork:3089:kewegner
Severity:1
Category:2
FormatWarning: 0
PrimaryMessage:The function handleSignal(int,int,CONTEXT*):608 reported
an error.
ExtendedMessage: A system error was detected - 40 SEG-VIOLATION (OS
signal nbr: 2).
Intel context flags:
i386/486 CONTROL INTEGER SEGMENT FULL FLOAT-POINT DEBUG-REG
Intel segment registers:
Gs:00000000 Fs:0000003b Es:00000023 Ds:00000023 Ss:00000023 Cs:0000001b
Intel GP registers:
Edi:068250f0 Esi:0225240f Ebx:068250f0 Edx:000009a4 Ecx:00000000
Eax:00000000 Ebp:02dbf42c Eip:064d61fa Esp:02dbf3fc
Error occurred at (Eip) :064d61fa, in source file: PolicyS.dll
Call stack unwind: base pointer (return address - DLL/EXE name)
02dbf42c (062C9120 - PolicyDB2.dll)
02dbf48c (06399054 - PolicyC.dll)
02dbf4c4 (0639998A - PolicyC.dll)
02dbf584 (06485190 - PolicyS.dll)
02dbf5d0 (6B6859E7 - somorori.dll)
02dbf828 (6B6857F8 - somorori.dll)
02dbfc0c (6EACDCE5 - somsrsai.dll)
02dbfd80 (6EACFB5D - somsrsai.dll)
02dbff54 (504419F7 - cppobi36.dll)
02dbffa4 (5043F663 - cppobi36.dll)
02dbffb8 (77F04F2C - KERNEL32.dll)
Module list: size (load address) date-linked-GMT => DLL/EXE name
512000 (00240000) 1998/04/18 04:27:22 GMT => c:\IBMCPPW\BIN\CPPWM35I.dll
19456 (00360000) 1998/09/09 10:46:19 GMT => e:\e9836.01.lite\lib.nt\teceifi.dll
289280 (00400000) 1998/09/09 11:22:59 GMT => e:\e9836.01.lite\bin.nt\somsrsm.exe
4605952 (004C0000) 1998/09/22 19:17:03 GMT => e:\e9836.01.lite\lib.nt\somibe1i.dll
5027328 (009B0000) 1998/09/16 18:47:26 GMT => e:\e9836.01.lite\lib.nt\somibs1i.dll
```

```
57856 (02490000) 1998/04/18 04:17:07 GMT => C:\IBMCPPW\LOCALE\iconv\UCSTBL.dll
16896 (030C0000) 1998/09/09 09:06:58 GMT => e:\e9836.01.lite\lib.nt\EN_US\somscl1i.dll
980992 (03350000) 1997/08/16 12:53:48 GMT => C:\SQLLIB\BIN\DB2APP.dll
1257472 (03450000) 1997/08/16 12:53:43 GMT => C:\SQLLIB\BIN\DB2SYS.dll
14848 (03590000) 1997/08/16 12:53:51 GMT => C:\SQLLIB\BIN\DB2WINT.dll
32768 (035A0000) 1997/08/16 12:53:44 GMT => C:\SQLLIB\BIN\DB2SYSP.dll
102400 (035B0000) 1997/08/16 12:53:37 GMT => C:\SQLLIB\BIN\DB2ABIND.dll
17920 (035D0000) 1997/08/16 12:55:59 GMT => C:\SQLLIB\BIN\DB2CAPRO.dll
37648 (03A10000) 1997/04/22 23:50:00 GMT => C:\WINNT\System32\rnr20.dll
34064 (04360000) 1997/04/25 19:33:47 GMT => C:\WINNT\System32\RpcLtScm.Dll
37648 (04670000) 1997/04/17 22:34:28 GMT => C:\WINNT\System32\RpcLtCcm.Dll
67072 (04A00000) 1997/04/25 23:13:52 GMT => C:\WINNT\System32\odbcint.dll
230672 (04A50000) 1997/04/25 23:13:44 GMT => C:\WINNT\System32\ODBC32.dll
4096 (05340000) 1998/09/09 10:09:57 GMT => e:\e9836.01.lite\lib.nt\db2slf.dll
893952 (06250000) 1998/09/09 11:08:23 GMT => e:\e9836.01.lite\lib.nt\PolicyDB2.dll
326656 (06380000) 1998/09/09 10:54:30 GMT => e:\e9836.01.lite\lib.nt\PolicyC.dll
2835456 (06430000) 1998/09/28 15:20:32 GMT => e:\e9836.01.lite\lib.nt\PolicyS.dll
1972224 (069B0000) 1998/09/09 11:21:11 GMT => e:\e9836.01.lite\lib.nt\JPolicyS.dll
414208 (06C10000) 1998/04/16 11:05:48 GMT => e:\e9836.01.lite\tools\nt\java\bin\javai.dll
46080 (09120000) 1998/04/16 11:39:15 GMT => e:\e9836.01.lite\tools\nt\java\bin\zip.dll
3240256 (10000000) 1998/01/16 22:10:53 GMT => C:\DCE\dcelocal\bin\libdce.dll
193024 (1C000000) 1998/01/12 19:34:43 GMT => C:\DCE\dcelocal\bin\pthreads.dll
162816 (28100000) 1997/09/15 15:58:32 GMT => e:\e9836.01.lite\lib.nt\setloc1.dll
519680 (50000000) 1997/09/15 18:00:13 GMT => c:\IBMCPPW\BIN\CPPWOB3I.dll
1283072 (50400000) 1997/11/05 22:05:46 GMT => e:\e9836.01.lite\tools\nt\ioc.t\bin\cppobi36.dll
2023424 (51000000) 1997/09/15 18:01:28 GMT => c:\IBMCPPW\BIN\CPPWOU3I.dll
13584 (5F810000) 1997/04/28 21:57:58 GMT => C:\WINNT\System32\rpcltc1.dll
313344 (68000000) 1998/09/09 13:02:41 GMT => e:\e9836.01.lite\lib.nt\somcscdi.dll
188416 (680E0000) 1998/09/09 13:02:41 GMT => e:\e9836.01.lite\lib.nt\somcscsi.dll
7168 (68170000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsdli.dll
32768 (681F0000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsmmi.dll
104960 (683C0000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcssni.dll
19968 (68460000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcssti.dll
6656 (68500000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsthi.dll
14336 (68580000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somcsuii.dll
794112 (68620000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somdc01i.dll
127488 (68760000) 1998/09/09 13:02:42 GMT => e:\e9836.01.lite\lib.nt\somdc02i.dll
396800 (688D0000) 1998/09/09 13:02:43 GMT => e:\e9836.01.lite\lib.nt\somdt00i.dll
24064 (68CD0000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somi2a1i.dll
289280 (68D60000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somi2s1i.dll
3622400 (68E10000) 1998/09/09 13:02:45 GMT => e:\e9836.01.lite\lib.nt\somiba1i.dll
210944 (691F0000) 1998/09/09 13:02:47 GMT => e:\e9836.01.lite\lib.nt\somibb1i.dll
855552 (69410000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibi1i.dll
166400 (69530000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibl1i.dll
373760 (695C0000) 1998/09/09 13:02:48 GMT => e:\e9836.01.lite\lib.nt\somibp1i.dll
175104 (69900000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somida1i.dll
58368 (699A0000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somide1i.dll
256512 (69A30000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somids1i.dll
659456 (69AF0000) 1998/09/09 13:02:49 GMT => e:\e9836.01.lite\lib.nt\somima1i.dll
50176 (69BF0000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somimb1i.dll
73728 (69C80000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somims1i.dll
510976 (6A1A0000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somira1i.dll
195584 (6A280000) 1998/09/09 13:02:50 GMT => e:\e9836.01.lite\lib.nt\somirs1i.dll
57344 (6A310000) 1998/09/09 13:02:51 GMT => e:\e9836.01.lite\lib.nt\somiym1i.dll
2354176 (6A3A0000) 1998/09/09 13:02:51 GMT => e:\e9836.01.lite\lib.nt\somlcm1i.dll
790528 (6A650000) 1998/09/09 13:02:52 GMT => e:\e9836.01.lite\lib.nt\somloc1i.dll
3214336 (6AF70000) 1998/09/09 13:02:56 GMT => e:\e9836.01.lite\lib.nt\somoq01i.dll
45568 (6B310000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somorbti.dll
182272 (6B3A0000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somorcdi.dll
693248 (6B4D0000) 1998/09/09 13:02:58 GMT => e:\e9836.01.lite\lib.nt\somoriri.dll
1086976 (6B660000) 1998/09/09 13:02:59 GMT => e:\e9836.01.lite\lib.nt\somorori.dll
173568 (6B9E0000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somorpti.dll
24064 (6BA90000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somorssi.dll
6035456 (6BB30000) 1998/09/09 13:03:00 GMT => e:\e9836.01.lite\lib.nt\somosb1i.dll
2080768 (6C150000) 1998/09/09 13:03:01 GMT => e:\e9836.01.lite\lib.nt\somosc1i.dll
1216512 (6C3C0000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmcii.dll
9728 (6C550000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmg1i.dll
```

```
1097216 (6C5F0000) 1998/09/09 13:03:02 GMT => e:\e9836.01.lite\lib.nt\sompmsii.dll
393216 (6CF10000) 1998/09/09 13:03:05 GMT => e:\e9836.01.lite\lib.nt\somrsbsi.dll
25088 (6D0F0000) 1998/09/09 13:03:06 GMT => e:\e9836.01.lite\lib.nt\somrsori.dll
125440 (6D180000) 1998/09/09 13:03:06 GMT => e:\e9836.01.lite\lib.nt\somrssri.dll
1385984 (6D440000) 1998/09/09 13:03:07 GMT => e:\e9836.01.lite\lib.nt\somscs1i.dll
638976 (6D600000) 1998/09/09 13:03:08 GMT => e:\e9836.01.lite\lib.nt\somsgcli.dll
709120 (6DAA0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somsgtsi.dll
48640 (6DBD0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somsh.dll
115200 (6DCC0000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somshcb.dll
99840 (6DD70000) 1998/09/09 13:03:09 GMT => e:\e9836.01.lite\lib.nt\somshcpi.dll
91648 (6DE00000) 1998/09/09 13:03:10 GMT => e:\e9836.01.lite\lib.nt\somshori.dll
20992 (6E290000) 1998/09/09 13:03:10 GMT => e:\e9836.01.lite\lib.nt\somsmdei.dll
160768 (6E4B0000) 1998/09/09 13:03:11 GMT => e:\e9836.01.lite\lib.nt\somsmfni.dll
101376 (6E790000) 1998/09/09 13:03:11 GMT => e:\e9836.01.lite\lib.nt\somsmoii.dll
52736 (6EA20000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsmsri.dll
965632 (6EA90000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsrsai.dll
671744 (6EBE0000) 1998/09/09 13:03:12 GMT => e:\e9836.01.lite\lib.nt\somsrssi.dll
2754560 (6EDA0000) 1998/09/09 13:03:13 GMT => e:\e9836.01.lite\lib.nt\somsss1i.dll
516096 (6F2F0000) 1998/09/09 13:03:17 GMT => e:\e9836.01.lite\lib.nt\somtrm1i.dll
2202112 (6F3F0000) 1998/09/09 13:03:17 GMT => e:\e9836.01.lite\lib.nt\somtrs1i.dll
2886144 (6F670000) 1998/09/09 13:03:18 GMT => e:\e9836.01.lite\lib.nt\somtrt1i.dll
427520 (6F9A0000) 1998/09/09 13:03:20 GMT => e:\e9836.01.lite\lib.nt\somtrx1i.dll
297984 (74C00000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\skit.dll
484352 (74C60000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\x509cms.dll
62976 (74CF0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\soedber.dll
34816 (74D10000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\soedapi.dll
13312 (74D50000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\ossdmem.dll
12288 (74D60000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\ossapi.dll
28160 (74D70000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\nspcommon.dll
18432 (74E00000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cstrain.dll
5120 (74E30000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cmskfra.dll
1566208 (74E50000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cms.dll
12288 (74FE0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\cmpval.dll
13824 (74FF0000) 1998/08/25 15:46:54 GMT => e:\e9836.01.lite\lib.nt\berreal.dll
44304 (77660000) 1997/04/17 22:34:19 GMT => C:\WINNT\system32\msafd.dll
70416 (77670000) 1997/04/22 23:49:50 GMT => C:\WINNT\System32\MSWSOCK.dll
18704 (77690000) 1997/03/28 01:58:49 GMT => C:\WINNT\System32\wshtcpip.dll
8464 (776A0000) 1997/01/02 19:54:11 GMT => C:\WINNT\System32\WS2HELP.dll
59664 (776B0000) 1997/04/03 01:10:00 GMT => C:\WINNT\System32\WS2_32.dll
20240 (776D0000) 1996/07/26 19:19:24 GMT => C:\WINNT\System32\WSOCK32.dll
41744 (777E0000) 1997/04/22 23:50:01 GMT => C:\WINNT\System32\SAMLIB.dll
224528 (77800000) 1997/04/25 19:33:39 GMT => C:\WINNT\System32\NETAPI32.dll
17168 (77840000) 1996/07/26 19:19:23 GMT => C:\WINNT\System32\NETRAP.dll
12560 (779C0000) 1996/07/26 19:19:21 GMT => C:\WINNT\system32\LZ32.dll
65024 (779D0000) 1996/07/26 19:19:21 GMT => C:\WINNT\System32\MSVCRT40.dll
36112 (77A90000) 1996/07/26 19:19:20 GMT => C:\WINNT\system32\VERSION.dll
704272 (77B20000) 1997/04/25 19:33:41 GMT => C:\WINNT\system32\ole32.dll
310032 (77BF0000) 1997/04/25 19:33:18 GMT => C:\WINNT\system32\COMCTL32.dll
1277200 (77C40000) 1997/04/25 19:33:49 GMT => C:\WINNT\system32\SHELL32.dll
185104 (77D80000) 1997/04/25 19:33:18 GMT => C:\WINNT\system32\comdlg32.dll
246544 (77DC0000) 1997/04/11 23:24:11 GMT => C:\WINNT\system32\ADVAPI32.dll
317200 (77E10000) 1997/04/28 21:57:58 GMT => C:\WINNT\system32\RPCRT4.dll
330512 (77E70000) 1997/04/25 19:33:52 GMT => C:\WINNT\system32\USER32.dll
165648 (77ED0000) 1997/04/25 19:33:25 GMT => C:\WINNT\system32\GDI32.dll
372496 (77F00000) 1997/04/25 19:33:31 GMT => C:\WINNT\system32\KERNEL32.dll
355088 (77F60000) 1997/04/11 20:38:50 GMT => C:\WINNT\System32\ntdll.dll
149264 (77FD0000) 1996/07/17 18:15:07 GMT => C:\WINNT\System32\WINMM.dll
271632 (78000000) 1997/01/23 07:07:13 GMT => C:\WINNT\system32\MSVCRT.dll
70656 (780A0000) 1997/01/23 05:27:47 GMT => C:\WINNT\System32\MSVCIRT.dll
Storage dump:Storage dump:
02DBF40C AA AA AA AA 40 AD 58 06 - AA AA AA AA AA AA AA AA ....@.X.........
02DBF41C 00 00 00 00 DC FF FF FF - 84 F4 DB 02 90 D0 53 06 ..............S.
02DBF42C 8C F4 DB 02 20 91 2C 06 - BC 55 82 06 0F 24 25 02 .....,..U...$%.
02DBF43C 3B 01 3D 06 F0 34 32 06 - 02 00 00 00 38 F4 DB 02 ;.=..42.....8...
02DBF44C DC BA 31 06 60 8A 82 06 - 30 42 30 06 00 00 00 00 ..1.'...0B0.....
02DBF45C E0 FF FF FF F0 B5 7B 06 - BC 55 82 06 AA 01 00 00 ......{..U......
02DBF46C A7 D7 F3 6C F0 B5 7B 06 - 01 8C 82 06 70 0B 82 06 ...l..{.....p...
```

```
02DBF47C 80 54 82 06 B8 FF FF FF - BC F4 DB 02 90 D0 2D 06 .T............-.
02DBF48C C4 F4 DB 02 54 90 39 06 - D4 3D 9A 06 3B 01 3D 06 ....T.9..=..;.=.
02DBF49C 01 00 00 00 A4 10 3D 06 - E4 15 3D 06 50 A8 F1 6E ......=...=.P..n
02DBF4AC 84 F5 DB 02 9C C9 67 6B - 60 52 82 06 D8 FF FF FF ......gk'R......
02DBF4BC 7C F5 DB 02 40 B1 3B 06 - 84 F5 DB 02 8A 99 39 06 |...@.;.......9.
02DBF4CC C0 89 82 06 D4 3D 9A 06 - 38 3E 9A 06 E0 38 25 02 .....=..8>...8%.
02DBF4DC 0C 00 00 00 D0 F4 DB 02 - 84 0F 3D 06 00 00 FD 00 ..........=.....
02DBF4EC 20 00 00 00 F0 AF CC 04 - 00 00 00 00 00 00 FD 00 ..............
02DBF4FC 08 8A 82 06 48 F5 DB 02 - 49 1C 25 00 00 B0 CC 04 ....H...I.%.....
02DBF50C A0 D8 23 06 78 D1 23 06 - 20 F8 DB 02 00 B0 CC 04 ..#.x.#........
02DBF51C 6B 16 41 50 40 00 00 00 - 34 72 52 50 00 00 00 00 k.AP@...4rRP....
—- END OF EXCEPTION DATA —-
RawDataLen:0
```

**RELATED REFERENCES**

"Chapter 2. Activity Log for Problem Determination" on page 3
"HandleSignal - Compilation Map File Example"

## HandleSignal - Compilation Map File Example

This is a sample map file for PolicyS.dll. Some information in this map file has been removed to compress the view. The notation, ***//*** **...**. identifies deleted information. The map file is divided into several sections. The sections that you would use to locate a method in a call stack are:

- **Image base**
  Identifies the starting image; if the load point is not specified on the link statement, the default will be 40000.
- **Publics by Value**
  Lists all the methods.

Here is the map file for PolicyS.dll:

```
IBM(R) Linker for Windows(R), Version 02.01.r2a_WTC355a Copyright (C) IBM
Corporation 1988, 1996. Copyright (C) Microsoft Corp. 1988-1989. All rights reserved.
Current option settings:
IBM(R) Linker for Windows(R), Version 02.01.r2a_WTC355a
Copyright (C) IBM Corporation 1988, 1996.
Copyright (C) Microsoft Corp. 1988-1989.
All rights reserved.
&127;
Current option settings:
&127;DLL&127;CODE:RX&127;DATA:RW
&127;Alignfile:0x00000200&127;ALIGNAddr:0x00010000&127;BASE:0x00400000
&127;MAXSIZE:0x10000000&127;BRowse&127;NODBgpack&127;DEbug
&127;DEFaultlibrarysearch&127;NOEXTdictionary&127;NOFIxed&127;NOFOrce
&127;HEAP:0x00100000,0x00001000&127;NOLinenumbers&127;NOLOgo
&127;MAP:PolicyS.map&127;OCache:0x00763400&127;NOOPTFunc&127;PMTYPE:VIO
&127;SEGMENTS:256 &127;STACK:0x00100000,0x00001000
&127;SUBSYSTEM:CONSOLE,03.10&127;NOVERBose
Image based at 00400000
POLICYS
Start Length Name Class —- 1 —-
00410000 0000000B4H .text CODE 32-bit
&127;at offset 00000000 00006H bytes from C:\IBMCPPW\lib\CPPWM35I.lib
&127;at offset 00000006 00006H bytes from C:\IBMCPPW\lib\CPPWM35I.lib
Origin Group
0000:0 FLAT
Address Publics by Name
004A6170 ?addBeneficiary__13PolicyBO_ImplFv
// ...
005B1FB8 ___vttQ2_9CosStream22StreamableProxyFactory
Address Publics by Value
```

```
00410000 ?_CRT_init
// ...
004A6148 ?premium__13PolicyBO_ImplFf
004A6170 ?addBeneficiary__13PolicyBO_ImplFv
004A6228 ?delBeneficiary__13PolicyBO_ImplFv
// ...
Program entry point at 0050D590
```

**RELATED REFERENCES**

“HandleSignal - General Page Fault (GPF) Exception” on page 11
“Activity Log Showing General Page Fault Exception” on page 161

# Appendix C. IBM Communication Server Trace Samples

This topic provides sample trace extracts from IBM Communications Server.

- "APPC Verb Trace Samples"
  - TP_STARTED (page 168)
  - SET_TP_PROPERTIES (page 169)
  - ALLOCATE (page 170)
  - SEND_DATA (page 172)
  - RECEIVE_AND_WAIT (page 173)
  - CONFIRMED (page 174)
  - GET_ATTRIBUTES (page 175)
  - CONFIRM (page 177)
  - SEND_ERROR
  - DEALLOCATE (page 177)
- "APPC I-Frames Trace Samples" on page 178
  - The Bind Request (page 179)
  - The Attach FMH-5 and Initial Send Data (page 180)
  - Application Data (page 180)
  - Exchange Log Names (XLN) (page 181)
  - FMH-7s (page 182)

For more information about reading Communications Server traces, which requires some knowledge of the SNA architecture, see the SNA library.

## APPC Verb Trace Samples

These are trace samples for the following APPC verbs:

- TP_STARTED (page 168)
- SET_TP_PROPERTIES (page 169)
- ALLOCATE (page 170)
- SEND_DATA (page 172)
- RECEIVE_AND_WAIT (page 173)
- CONFIRMED (page 174)
- GET_ATTRIBUTES (page 175)
- CONFIRM (page 177)
- SEND_ERROR
- DEALLOCATE (page 177)

When reading the trace samples, refer to the following for additional information on opcodes, verb parameters, and return codes:

- “IBM Communication Server APPC Interface: Operation Codes” on page 185 for example, opcode such as fill and what_rcvd.
- “IBM Communication Server APPC Interface: Verb Parameters” on page 194 for example, conv_type, sync_level, rtn_ctl, security, and type.
- “IBM Communication Server APPC Interface: Return Codes” on page 195 for example, primary_rc and secondary_rc.

The notation //... means that text has been removed in the sample trace.

**TP_STARTED**
The TP_STARTED verb starts a local transaction program (TP). This is used to group a number of related conversations together. For example, all of the conversations that belong to the same Component Broker transaction will run under the same local TP. Values of interest in these trace entries are the **tp_id** and local **tp_name**. The **tp_id** is passed on all conversation requests so it is useful for grouping related conversations together. The local **tp_name** shows the local LU name in used by the server for this local TP.

```
_____________________________________________
[57] 07/08 16:41:38.27,(0082) len=188, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
TP_STARTED :
opcode                          = 14
opext                           = 0
format                          = 0
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
lu_alias[8]                     = 41554937444A3031
                                  . . . . . . . .
                                  A U I 7 D J 0 1
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
tp_name[64]                     = 43425365727665723A555349424D4E522E41554//...
                                  . . . . . . . . . . . . . . ( + . . . . .//...
                                  C B S e r v e r : U S I B M N R . A U I//...
delay_start                     = f2
enable_pool                     = 4b
pip_dlen                        = c9
_____________________________________________
[58] 07/08 16:41:38.27,(0083) len=188, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
TP_STARTED :
opcode                          = 14
opext                           = 0
format                          = 0
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
lu_alias[8]                     = 41554937444A3031
                                  . . . . . . . .
                                  A U I 7 D J 0 1
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
tp_name[64]                     = 43425365727665723A555349424D4E522E4155//...
                                  . . . . . . . . . . . . . . ( + . . . . //...
                                  C B S e r v e r : U S I B M N R . A U //...
```

```
delay_start                     = f2
enable_pool                     = 4b
pip_dlen                        = c9
```

### SET_TP_PROPERTIES

The SET_TP_PROPERTIES verb is used to associate two logical unit of work identifiers (LUWIds) with the local TP. One of the LUWIds (see `unprot_id`) is used on conversations that not recoverable (synclevel 0 or synclevel 1) and the other (`prot_id`) is used on recoverable (synclevel 2) conversations. LUWIds can be thought of in their simplest terms as the SNA version of the transaction identifier. The critical LUWId is the one for recoverable conversations as this is required to relate all updates for a particular Component Broker transaction together. The LUWId for non-recoverable conversations is used for accounting purposes.

```
[59] 07/08 16:41:38.28,(0084) len=196, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
SET_TP_PROPERTIES :
opcode                          = 7e
opext                           = 0
format                          = 0
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
set_prot_id                     = 1
new_prot_id                     = 0
prot_id.fq_lu_name_len          = 10
prot_id.fq_luw_name[17]         = E4E2C9C2D4D5D94BC1E4C9F7C4D1F0F18A
                                  U S I B M N R . A U I 7 D J 0 1 .
                                  . . . . . . . K . . . . . . . . .
prot_id.instance[6]             = E7A335000000
                                  X t . @ @ @
                                  . . 5 . . .
prot_id.sequence[2]             = 0100
                                  . @
                                  . .
set_unprot_id                   = 1
new_unprot_id                   = 0
unprot_id.fq_lu_name_len        = 10
unprot_id.fq_luw_name[17]       = E4E2C9C2D4D5D94BC1E4C9F7C4D1F0F192
                                  U S I B M N R . A U I 7 D J 0 1 k
                                  . . . . . . . K . . . . . . . . .
unprot_id.instance[6]           = E7A335020000
                                  X t . . @ @
                                  . . 5 . . .
unprot_id.sequence[2]           = 0100
                                  . @
                                  . .
set_user_id                     = 0
set_password                    = 0
user_id[10]                     = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
new_password[10]                = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
```

The exit trace entry just confirms whether the request was successful or not.

```
_____________________________________________
[60] 07/08 16:41:38.28,(0085) len=196, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
SET_TP_PROPERTIES :
opcode                          = 7e
opext                           = 0
format                          = 0
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
set_prot_id                     = 1
new_prot_id                     = 0
prot_id.fq_lu_name_len          = 10
prot_id.fq_luw_name[17]         = E4E2C9C2D4D5D94BC1E4C9F7C4D1F0F18A
                                  U S I B M N R . A U I 7 D J 0 1 .
                                  . . . . . . . K . . . . . . . . .
prot_id.instance[6]             = E7A335000000
                                  X t . @ @ @
                                  . . 5 . . .
prot_id.sequence[2]             = 0100
                                  . @
                                  . .
set_unprot_id                   = 1
new_unprot_id                   = 0
unprot_id.fq_lu_name_len        = 10
unprot_id.fq_luw_name[17]       = E4E2C9C2D4D5D94BC1E4C9F7C4D1F0F192
                                  U S I B M N R . A U I 7 D J 0 1 k
                                  . . . . . . . K . . . . . . . . .
unprot_id.instance[6]           = E7A335020000
                                  X t . . @ @
                                  . . 5 . . .
unprot_id.sequence[2]           = 0100
                                  . @
                                  . .
set_user_id                     = 0
set_password                    = 0
user_id[10]                     = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
new_password[10]                = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
_____________________________________________
```

### ALLOCATE

The ALLOCATE verb is used to create a new conversation using information from the PAA APPC Connection. This information is build into an attach FMH-5 which can be seen in the I-Frames trace.

```
_____________________________________________
[61] 07/08 16:41:38.28,(0086) len=284, APPN and APPC.APPC API.0000, 00000000://..
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
ALLOCATE :
opcode                          = 1
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
```

```
                                . @ . . @ @ @ @
                                . . . . . . . .
conv_id                         = 0
conv_type                       = 1
sync_level                      = 2
rtn_ctl                         = 0
duplex_type                     = 0
conv_group_id                   = 0
sense_data                      = 0
plu_alias[8]                    = 53544C4C55312020
                                  . . << . . . .
                                  S T L L U 1
mode_name[8]                    = D3F6F2D4C4C5F0F1
                                  L 6 2 M D E 0 1
                                  . . . . . . . .
tp_name[64]                     = C9E5E3D5D6404040404040404//...
                                   . . . . . @ @ @ @ @ @ @ //...
security                        = 0
pwd[10]                         = 2A2A2A2A2A2A2A2A2A2A
                                  . . . . . . . . . .
                                  * * * * * * * * * *
user_id[10]                     = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
pip_dlen                        = 0
pip_dptr                        = 0
fqplu_name[17]                  = 00000000000000000000000000000000000
                                  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . . . . . . . . .
_____________________________________________
```

The exit trace shows the **conv_id** which is an identifier for the conversation. This will be passed on all subsequent requests that relate to this conversation.

```
_____________________________________________
[62] 07/08 16:41:38.28,(0087) len=284, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
ALLOCATE :
opcode                          = 1
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
conv_id                         = 2fd0001
conv_type                       = 1
sync_level                      = 2
rtn_ctl                         = 0
duplex_type                     = 0
conv_group_id                   = 2d30001
sense_data                      = 0
plu_alias[8]                    = 53544C4C55312020
                                  . . << . . . .
                                  S T L L U 1
mode_name[8]                    = D3F6F2D4C4C5F0F1
                                  L 6 2 M D E 0 1
                                  . . . . . . . .
tp_name[64]                     = C9E5E3D5D640404040404040//...
                                  I V T N O
                                  . . . . . @ @ @ @ @ @ @ //...
security                        = 0
pwd[10]                         = 2A2A2A2A2A2A2A2A2A2A
```

```
                                  . . . . . . . . . .
                                  * * * * * * * * * *
user_id[10]                     = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
pip_dlen                        = 0
pip_dptr                        = 0
fqplu_name[17]                  = 00000000000000000000000000000000000
                                  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . . . . . . . . .
```

**SEND_DATA**

The SEND_DATA verb is used to pass GDS Records and Presentation Services Headers from Component Broker to IBM Communication Server so they can be sent to the tier-3 system. The example below shows data that makes up a PAO CRUD method request which has been added to a GDS record with an ID field of **0x12FF**.

```
[63] 07/08 16:41:38.30,(0088) len=185, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
SEND_DATA :
opcode                          = f
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
conv_id                         = 2fd0001
rts_rcvd                        = 0
expd_rcvd                       = 0
dlen                            = 35               < == This length is in hex
dptr                            = 82b01600
type                            = 0
Data Area 1:               Length = 53             < == This length is in decimal
  0035 12FF 40404040 C4C9E2D7 D3C1E840  < .5..@@@@.......@ >  < ....    DISPLAY  >
  D3C1E2E3 F1404040 40404040 40404040  < .....@@@@@@@@@@@@ >  < LAST1            >
  40404040 40404040 40404040 40404040  < @@@@@@@@@@@@@@@@ >  <                  >
  40404040 40                          < @@@@@            >  <                  >
```

The following example is a GDS record with an Id of **0x1211** which is an exchange log names record.

```
Data Area 1:               Length = 58
  003A 1211 026010E4 E2C9C2D4 D5D94BC1  < .:.../........K. >  < .....-.USIBMNR.A >
  E4C9F7C4 D1F0F122 C3C27AA2 9694A399  < .......".."z..... >  < UI7DJ01.CB:somtr >
  7AA385A2 A3A299A5 7AF3F5C1 F3C5F7F8  < z.......z....... >  < :testsrv:35A3E78 >
  F37AF0F0 F0C3F7F0 C2F3               < .z........       >  < 3:000C70B3       >
```

If the LL value is **0x0001**, then this is a Presentation Service (PS) header used for flowing commit messages between Component Broker and tier-3 systems.

```
Data Area 1:               Length = 8
  0001 060A 00050001                   < ........         >  < ........         >
```

The return trace entry just confirms whether the send was successful.

```
[64] 07/08 16:41:38.30,(0089) len=185, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)            Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
SEND_DATA :
opcode                           = f
opext                            = 0
format                           = 1
primary_rc                       = 0  OK
secondary_rc                     = 0  OK
tp_id[8]                         = 0100EE0200000000
                                   . @ . . @ @ @ @
                                   . . . . . . . .
conv_id                          = 2fd0001
rts_rcvd                         = 0
expd_rcvd                        = 0
dlen                             = 35
dptr                             = 82b01600
type                             = 0
Data Area 1:                Length = 53
  003512FF 40404040 C4C9E2D7 D3C1E840  < .5..@@@@.......@ >  < ....    DISPLAY  >
  D3C1E2E3 F1404040 40404040 40404040  < .....@@@@@@@@@@@ >  < LAST1            >
  40404040 40404040 40404040 40404040  < @@@@@@@@@@@@@@@@ >  <                  >
  40404040 40                          < @@@@@            >  <                  >
```

### RECEIVE_AND_WAIT

The RECEIVE_AND_WAIT call is used for receiving data from the network. The entry trace shows the maximum amount of data that can be received (see max_len). The buffer into which the data is to be received is shown in this trace entry but its contents are not significant. So it may show uninitialized storage, of the data sent/received by a previous request.

```
[65] 07/08 16:41:38.30,(008A) len=237, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)            Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
RECEIVE_AND_WAIT :
opcode                           = b
opext                            = 0
format                           = 1
primary_rc                       = 0  OK
secondary_rc                     = 0  OK
tp_id[8]                         = 0100EE0200000000
                                   . @ . . @ @ @ @
                                   . . . . . . . .
conv_id                          = 2fd0001
what_rcvd                        = 0
rtn_status                       = 1
fill                             = 0
rts_rcvd                         = 0
expd_rcvd                        = 0
max_len                          = 5d
dlen                             = 0
dptr                             = 82b01600
Data Area 1:                Length = 93
  AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA  < ................ >  < ................ >
  AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA  < ................ >  < ................ >
  AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA  < ................ >  < ................ >
  AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA  < ................ >  < ................ >
```

```
  AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA  < ................ >  < ................ >
  AAAAAAAA AAAAAAAA AAAAAAAA AA        < .............    >  < .............    >
```

The exit trace is more interesting as it shows the actual data received and the accompanying indicators. See “IBM Communication Server APPC Interface: Operation Codes” on page 185.

```
[71] 07/08 16:41:39.75,(0090) len=237, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)            Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
RECEIVE_AND_WAIT :
opcode                           = b
opext                            = 0
format                           = 1
primary_rc                       = 0  OK
secondary_rc                     = 0  OK
tp_id[8]                         = 0100EE0200000000
                                   . @ . . @ @ @ @
                                   . . . . . . . .
conv_id                          = 2fd0001
what_rcvd                        = 103
rtn_status                       = 1
fill                             = 0
rts_rcvd                         = 0
expd_rcvd                        = 0
max_len                          = 5d
dlen                             = 5d
dptr                             = 82b01600
Data Area 1:                Length = 93
  005D12FF C5D5E3D9 E840E6C1 E240C4C9  < .]....... @...@.. >  < .)..ENTRY WAS DI >
  E2D7D3C1 E8C5C440 40404040 40404040  < ....... @@@@@@@@@@ >  < SPLAYED          >
  40404040 40404040 40404040 C4C9E2D7  < @@@@@@@@@@@@.... >  <             DISP >
  D3C1E840 D3C1E2E3 F1404040 4040C6C9  < ...@.....@@@@@.. >  < LAY LAST1     FI >
  D9E2E3F1 40404040 F860F1F1 F160F1F1  < ....@@@@.'...'.. >  < RST1    8-111-11 >
  F1F1C4F0 F161D9F0 F1F0F0F0 F1        < .....a.......    >  < 11D01/R010001    >
```

If this is a receive for application data (ID = 0x12FF), then you would expect that `max_len`, `dlen` and the LL would be the same value (0x5d in this example). If they are not, then there is a mismatch between the output buffer copy book used in CICON to generate the PAO and the real application in the tier-3 system.

**CONFIRMED**

The CONFIRMED verb is used to respond positively to a RQD2 indicator. See “Transmission Headers (TH) and Request/Response Headers (RH)” on page 207 for more information on RQD2.

```
[53] 07/08 16:41:38.25,(007E) len=120, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)            Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
CONFIRMED :
opcode                           = 4
opext                            = 0
format                           = 1
primary_rc                       = 0  OK
secondary_rc                     = 0  OK
tp_id[8]                         = 0400E30100000000
                                   . @ T . @ @ @ @
                                   . . . . . . . .
```

```
conv_id                         = 1ea0004
_____________________________________________
[55] 07/08 16:41:38.25,(0080) len=120, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
CONFIRMED :
opcode                          = 4
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0400E30100000000
                                  . @ T . @ @ @ @
                                  . . . . . . . .
conv_id                         = 1ea0004
_____________________________________________
```

### GET_ATTRIBUTES

The GET_ATTRIBUTES verb is used by the Component Broker server to extract the conversation correlator and session identifier for the conversation. The entry trace is not very useful.

```
_____________________________________________
[87] 07/08 16:41:48.32,(00A0) len=28, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       400021031054 (canonical: 020084c0082a)
Source SAP:           04
Destination Address:  000629f0d8fa (canonical: 0060940f1b5f)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c80e6419641f001f2f1d3b0
LPDU Type:            RR
Command Flag:         R              Poll Flag:            0
NR:                   0c
LPDU Data length:     0
[88] 07/08 16:41:51.32,(00A1) len=260, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
GET_ATTRIBUTES :
opcode                          = 7
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
conv_id                         = 2fd0001
sync_level                      = 0
mode_name[8]                    = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
net_name[8]                     = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
lu_name[8]                      = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
lu_alias[8]                     = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
plu_alias[8]                    = 0000000000000000
                                  @ @ @ @ @ @ @ @
```

```
                                  . . . . . . . .
plu_un_name[8]                  = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
fqplu_name[17]                  = 00000000000000000000000000000000
                                  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . . . . . . . .
user_id[10]                     = 00000000000000000000
                                  @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . .
conv_group_id                   = 0
conv_corr_len                   = 0
conv_corr[8]                    = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
luw_id.fq_lu_name_len           = 0
luw_id.fq_luw_name[17]          = 00000000000000000000000000000000
                                  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
                                  . . . . . . . . . . . . . . . .
luw_id.instance[6]              = 000000000000
                                  @ @ @ @ @ @
                                  . . . . . .
luw_id.sequence[2]              = 0000
                                  @ @
                                  . .
sess_id[8]                      = 0000000000000000
                                  @ @ @ @ @ @ @ @
                                  . . . . . . . .
_____________________________________________
```

However, in addition to the conversation correlator (see conv_corr) and session identifier (see sess_id), the return trace entry summarizes all of the information known about the conversation.

```
_____________________________________________
[89] 07/08 16:41:51.32,(00A2) len=260, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
GET_ATTRIBUTES :
opcode                          = 7
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100EE0200000000
                                  . @ . . @ @ @ @
                                  . . . . . . . .
conv_id                         = 2fd0001
sync_level                      = 2
mode_name[8]                    = D3F6F2D4C4C5F0F1
                                  L 6 2 M D E 0 1
                                  . . . . . . . .
net_name[8]                     = E4E2C9C2D4D5D940
                                  U S I B M N R
                                  . . . . . . . @
lu_name[8]                      = C1E4C9F7C4D1F0F1
                                  A U I 7 D J 0 1
                                  . . . . . . . .
lu_alias[8]                     = 41554937444A3031
                                  . . . . . . . .
                                  A U I 7 D J 0 1
plu_alias[8]                    = 53544C4C55312020
                                  . . < < . . . .
                                  S T L L U 1
plu_un_name[8]                  = E2E3E8F7C9D4F1F6
```

```
                                  S T Y 7 I M 1 6
                                  . . . . . . . .
fqplu_name[17]                  = E4E2C9C2D4E2E3E84BE2E3E8F7C9D4F1F6
                                  U S I B M S T Y . S T Y 7 I M 1 6
                                  . . . . . . . . K . . . . . . . .
user_id[10]                     = 40404040404040404040
                                  @ @ @ @ @ @ @ @ @ @
conv_group_id                   = 2d30001
conv_corr_len                   = 8
conv_corr[8]                    = 0100FC02D8D9C4E1
                                  . @ . . Q R D .
                                  . . . . . . . .
luw_id.fq_lu_name_len           = 10
luw_id.fq_luw_name[17]          = E4E2C9C2D4D5D94BC1E4C9F7C4D1F0F18A
                                  U S I B M N R . A U I 7 D J 0 1 .
                                  . . . . . . . K . . . . . . . . .
luw_id.instance[6]              = E7A335000000
                                  X t . @ @ @
                                  . . 5 . . .
luw_id.sequence[2]              = 0100
                                  . @
                                  . .
sess_id[8]                      = D93B11F565630EEF
                                  R . . 5 . . . .
                                  . ; . . e c . .
_____________________________________________
```

The Conversation Correlator and Session Id are used to build the Compare States GDS Record during recovery.

**SEND_ERROR**
The SEND_ERROR call (in conjunction with the CONFIRM call is used by Component Broker to send a backout request.

**CONFIRM**
The CONFIRM call (in conjunction with the SEND_ERROR call is used by Component Broker to send a backout request.

**DEALLOCATE**
The DEALLOCATE call ends a conversation. The dealloc_type parameter indicates how the conversation is to be terminated.

```
_____________________________________________
[63] 07/13 14:58:33.92,(00AC) len=140, APPN and APPC.APPC API.0000, 00000000://...
API Name    - APPC(K)           Entry Point - APPC
Process ID  - 0x       0
Usage       - Entry
Trace Level is DATA
DEALLOCATE :
opcode                          = 5
opext                           = 0
format                          = 1
primary_rc                      = 0  OK
secondary_rc                    = 0  OK
tp_id[8]                        = 0100624E00000000
                                  . @ . + @ @ @ @
                                  . . b N . . . .
conv_id                         = 4e660001
expd_rcvd                       = 0
dealloc_type                    = 3
log_dlen                        = 0
log_dptr                        = 0
implied_forget                  = 0
_____________________________________________
[66] 07/13 14:58:33.92,(00AF) len=128, APPN and APPC.APPC API.0000, 00000000://...
```

```
API Name    - APPC(K)            Entry Point - APPC
Process ID  - 0x       0
Usage       - Exit
Trace Level is DATA
DEALLOCATE :
opcode                           = 5
opext                            = 0
format                           = 1
primary_rc                       = 0  OK
secondary_rc                     = 0  OK
tp_id[8]                         = 0100624E00000000
                                   . @ . + @ @ @ @
                                   . . b N . . . .
conv_id                          = 4e660001
expd_rcvd                        = 0
dealloc_type                     = 3
log_dlen                         = 0
log_dptr                         = 0
implied_forget                   = 0
_____________________________________________
```

**RELATED REFERENCES**


“APPC I-Frames Trace Samples”
“Appendix F. SNA Data Formats” on page 207
“Transmission Headers (TH) and Request/Response Headers (RH)” on page 207
“Attach FMH-5” on page 209
“Logical Unit of Work Identifier (LUWId)” on page 210
“GDS Records and Data Mapping” on page 210
“Presentation Services (PS) Header 10” on page 212
“Function Management Header 7 (FMH-7)” on page 213
“Exchange Log Names (XLN) GDS Record” on page 214
“Compare States GDS Record” on page 215
“Conversation Correlator (CC) and Session Id” on page 216


## APPC I-Frames Trace Samples

When you request an I-Frames trace from IBM Communications Server, entries similar to the one shown below are written to the trace file. They contain the actual SNA records sent over the network, and you need to have some SNA knowledge to interpret them.

The notation //... means that text has been removed in the sample trace.

The following example shows the I-Frames entry. Each entry contains a Transmission Header (TH), a Request/Response Header (RH), and a Request/Response Unit (RU). In the example, the TH is 2E000102 0001, the RH is 0B91 20, and the RU is the rest of the block of data.

```
[47] 07/08 16:41:37.50,(0078) len=143, Connectivity.LAN (LLC2).0001, 00000000://...
000000
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c00e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   06
NS:                   04
LPDU Data length:     115
  2E000102 0001 0B91 20300502 FF0003D0  < ........ 0..... >  < .......j.......} >
  00400206 F2001910 E4E2C9C2 D4D5D94B  < .@.............K >  < . ..2...USIBMNR. >
  C1E4C9F7 C4D1F0F1 8BE7A335 01000001  < ...........5.... >  < AUI7DJ01.Xt..... >
  080400EB 01ACDBC4 E1003A12 11026010  < ..........:...'. >  < .......D......-. >
  E4E2C9C2 D4D5D94B C1E4C9F7 C4D1F0F1  < .......K........ >  < USIBMNR.AUI7DJ01 >
  22C3C27A A29694A3 997AA385 A2A3A299  < "..z.....z...... >  < .CB:somtr:testsr >
  A57AF3F5 C1F3C5F7 F8F37AF0 F0F0C3F7  < .z........z..... >  < v:35A3E783:000C7 >
  F0C2F3                               < ...             >  < 0B3             >
```

In general, the interesting data is in the RU which contains the data sent between the Component Broker Server and tier-3 system. Some examples of the more common I-Frames trace entries are shown below.

**The Bind Request**

In an I-Frames trace, there are often many entries similar to the one below. These are bind requests which request that a session is started between two LUs. Look for the **0x6** and **0x31** shown in **bold** to identify a bind request.

```
[29] 07/08 16:41:31.36,(0066) len=190, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c00e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   00
NS:                   00
LPDU Data length:     162
  2F000002 8001 6B81 00 31 0013 07B0B050  < /.....k..1.....P >  < ......,a.......& >
  33018186 86810106 02000000 00000000  < 3.............. >  < ..affa.......... >
  16430000 10E4E2C9 C2D4D5D9 4BC1E4C9  < .C..........K... >  < .....USIBMNR.AUI >
  F7C4D1F0 F1320009 02E2D5C1 E2E5C3D4  < .....2.......... >  < 7DJ01....SNASVCM >
  C7090301 3B11F565 630EEE11 04E4E2C9  < ....;..ec........ >  < G.....5......USI >
  C2D4D5D9 4BC1E4C9 F7C4D1F0 F10A1300  < ....K........... >  < BMNR.AUI7DJ01... >
  622D655C 6A743F6E 0011E4E2 C9C2D4E2  < b-e\jt?n........ >  < ...*|..>..USIBMS >
  E3E84BE2 E3E8F7C9 D4F1F62C 0A0408E2  < ..K.......,.... >  < TY.STY7IM16....S >
  D5C1E2E5 C3D4C760 19D93B11 F565630E  < .......'..;..ec. >  < NASVCMG-.R..5... >
  EE10E4E2 C9C2D4D5 D94BC1E4 C9F7C4D1  < .........K...... >  < ..USIBMNR.AUI7DJ >
  F0F0                                 < ..              >  < .00             >
```

If a bind is successful, you will see a bind response coming back. Again, the important values to look for are in **bold**. Successful binds mean that basic communication is possible.

```
[32] 07/08 16:41:34.27,(0069) len=135, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       400021031054 (canonical: 020084c0082a)
Source SAP:           04
Destination Address:  000629f0d8fa (canonical: 0060940f1b5f)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c80e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:           0
NR:                   01
NS:                   01
LPDU Data length:     107
  2F000200 8001EB80 00310013 07B0B050  < /........1.....P >  < ..............& >
  B3008085 85800006 02000000 00000000  < ................ >  < ...ee........... >
  10430000 00280009 02E2D5C1 E2E5C3D4  < .C...(.......... >  < .........SNASVCM >
  C7090302 3B11F565 630EEE12 05E4E2C9  < ....;..ec........ >  < G.....5......USI >
  C2D4E2E3 E84BE2E3 E8F7C9D4 F1F60000  < .....K.......... >  < BMSTY.STY7IM16.. >
  6019D93B 11F56563 0EEE10E4 E2C9C2D4  < '..;..ec........ >  < -.R..5.....USIBM >
  D5D94BC1 E4C9F7C4 D1F0F0             < ..K........      >  < NR.AUI7DJ00      >
```

### The Attach FMH-5 and Initial Send Data

When a Component Broker server wants to send data to a tier-3 system, it requires an APPC conversation. These are created using an ALLOCATE request. Information passed on the ALLOCATE is packed into an attach FMH-5 record. An example is shown below in **bold**. Search for the **0502 FF** pattern to locate it in the trace. Following the attach FMH-5 in the I-Frames trace is the first data sent on the conversation. This is formatted into a GDS record. The GDS record is highlighted in *italics* below. Further examples of GDS records can be found in application data and Exchange Log Names (XLN).

```
[33] 07/08 16:41:34.27,(006A) len=110, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c00e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:           0
NR:                   02
NS:                   01
LPDU Data length:     82
  2E000002 00010B91 2030 0502 FF0003D0  < ........ 0...... >  < .......j.......} >
  00000206 F1001910 E4E2C9C2 D4D5D94B  < ..............K >  < ....1...USIBMNR. >
  C1E4C9F7 C4D1F0F1 B1B8C841 4D490001  < ...........AMI.. >  < AUI7DJ01..H.(... >
  080400ED 01E4DCC4 E1 001912 10020000  < ................ >  < .....U.D........ >
  00000020 00100000 0008D3F6 F2D4C4C5  < ... ............ >  < ..........L62MDE >
  F0F1                                 < ..             >  < 01             >
```

### Application Data

Application data (that is, the data that makes up the PAO CRUD method requests and responses) is formatted into GDS records with an ID of **0x12FF**.

```
[81] 07/08 16:41:46.54,(009A) len=141, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c00e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   0b
NS:                   0a
LPDU Data length:     113
  2E000202 00010B91 20330502 FF0003D1  < ........ 3..... >  < .......j.......J >
  008005C9 E5E3D5D6 001910E4 E2C9C2D4  < ................ >  < ...IVTNO...USIBM >
  D5D94BC1 E4C9F7C4 D1F0F18A E7A33500  < ..K...........5. >  < NR.AUI7DJ01.Xt.. >
  00000108 0100F602 D8D9C4E1 003512FF  < .............5.. >  < ......6.QRD..... >
  40404040 E4D7C4C1 E3C54040 D3C1E2E3  < @@@@......@@.... >  <   UPDATE  LAST>
  F1404040 4040C2C1 D9D5C5E8 40404040  < .@@@@@......@@@@ >  < 1     BARNEY     >
  F3F0F9F0 F9404040 4040C5D4 E260F8F6  < .....@@@@@...'.. >  < 30909     EMS-86 >
  F2                                   < .              >  < 2               >
```

```
[84] 07/08 16:41:48.11,(009D) len=130, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       400021031054 (canonical: 020084c0082a)
Source SAP:           04
Destination Address:  000629f0d8fa (canonical: 0060940f1b5f)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c80e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   0b
NS:                   0c
LPDU Data length:     102
  2E000202 000103A1 20005D12 FFC5D5E3  < ........ .]..... >  < ........~..)..ENT >
  D9E840E6 C1E240E4 D7C4C1E3 C5C44040  < ..@...@.......@@ >  < RY WAS UPDATED   >
  40404040 40404040 40404040 40404040  < @@@@@@@@@@@@@@@@ >  <                  >
  40404040 40E4D7C4 C1E3C540 40D3C1E2  < @@@@@......@@... >  <      UPDATE  LAS >
  E3F14040 404040C2 C1D9D5C5 E8404040  < ..@@@@@......@@@ >  < T1     BARNEY    >
  40F3F0F9 F0F94040 404040C5 D4E260F8  < @.....@@@@@...'. >  <  30909     EMS-8 >
  F6F2F0F0 F0F1                        < ......         >  < 620001          >
```

### Exchange Log Names (XLN)

The exchange log names (XLN) process is used by two SNA systems to make sure they are both using the correct transaction log. One side (either the Component Broker server or the tier-3 system) requests a conversation with the other system's 0x06F2 transaction. It then sends an exchange log names (XLN) GDS record (which may be be followed by a Compare States GDS Record if there is an outstanding transaction to resolve) and issues a receive to pick up the response. This response will contain the response XLN GDS record (and possibly a response Compare States GDS Record). The conversation is then terminated using CONFIRM/CONFIRMED.

```
[47] 07/08 16:41:37.50,(0078) len=143, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c00e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   06
NS:                   04
LPDU Data length:     115
  2E000102 00010B91 20300502 FF0003D0  < ........ 0...... >  < .......j.......} >
  00400206 F2001910 E4E2C9C2 D4D5D94B  < .@.............K >  < . ..2...USIBMNR. >
  C1E4C9F7 C4D1F0F1 8BE7A335 01000001  < ...........5.... >  < AUI7DJ01.Xt..... >
  080400EB 01ACDBC4 E1003A12 11026010  < .........:...'. >  < .......D......-. >
  E4E2C9C2 D4D5D94B C1E4C9F7 C4D1F0F1  < .......K........ >  < USIBMNR.AUI7DJ01 >
  22C3C27A A29694A3 997AA385 A2A3A299  < "..z.....z...... >  < .CB:somtr:testsr >
  A57AF3F5 C1F3C5F7 F8F37AF0 F0F0C3F7  < .z........z..... >  < v:35A3E783:000C7 >
  F0C2F3                               < ...             >  < 0B3             >
```

```
[51] 07/08 16:41:38.14,(007C) len=40, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       000629f0d8fa (canonical: 0060940f1b5f)
Source SAP:           04
Destination Address:  400021031054 (canonical: 020084c0082a)
Destination SAP:      04
RIF Length:           12
RIF Data:             0c80e6419641f001f2f1d3b0
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   05
NS:                   07
LPDU Data length:     58
  2E000201 000103A1 01003112 11096011  < .........1...'. >  < .......~......-. >
  E4E2C9C2 D4E2E3E8 4BE2E3E8 F7C9D4F1  < ........K....... >  < USIBMSTY.STY7IM1 >
  F618C1E3 D94BC2F0 C2F8C2F7 F1F3C6F1  < .....K.......... >  < 6.ATR.B0B8B713F1 >
  C5C2F3F6 F0F24BC9 C2D4               < ......K...       >  < EB3602.IBM       >
```

**Function Management Header (FMH-7)**

The FMH-7 is a record used for sending errors, backout requests and for abnormally terminating conversations. The type of error it represents is given in the four-byte sense code. This is **0x08640001** in the example below.

```
[65] 07/13 14:58:33.92,(00AE) len=42, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       0004acf7a815 (canonical: 002035ef15a8)
Source SAP:           04
Destination Address:  400045121088 (canonical: 0200a2480811)
Destination SAP:      04
RIF Length:           10
RIF Data:             0a30028a700a00310060
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   0d
NS:                   28
LPDU Data length:     16
  2E000202 003E0B80 01070708 64000100  < .....>......d... >  < ................ >
```

A Component Broker server will use a SEND_ERROR verb followed by a CONFIRM verb to send an FMH-7 that represents a backout request and a Request Definite Response (RQD2). It will use a DEALLOCATE request with the

appropriate dealloc_type parameter (which selects the sense code) to send an FMH-7 with a Conditional End Bracket (CEB) to abnormally terminate a conversation.

When not sending application data, it is possible to request that an FMH-7 be sent in Receive state. When this occurs the FMH-7 is preceded by a negative response **0x0846** (see below) to make the partner system send the Change Direction (CD).

See “Transmission Headers (TH) and Request/Response Headers (RH)” on page 207 for more information on RQD2, CD, and CEB.

```
[64] 07/13 14:58:33.92,(00AD) len=39, Connectivity.LAN (LLC2).0001, 00000000://...
Frame Type:           TOKEN_RING
Source Address:       0004acf7a815 (canonical: 002035ef15a8)
Source SAP:           04
Destination Address:  400045121088 (canonical: 0200a2480811)
Destination SAP:      04
RIF Length:           10
RIF Data:             0a30028a700a00310060
LPDU Type:            I-Frame
Command Flag:         C              Poll Flag:            0
NR:                   0d
NS:                   27
LPDU Data length:     13
  2E000202 802D8730 00084600 00       < .....-.0..F..    >  < ......g......    >
```

**RELATED REFERENCES**

“Appendix C. IBM Communication Server Trace Samples” on page 167
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207
“Transmission Headers (TH) and Request/Response Headers (RH)” on page 207
“Attach FMH-5” on page 209
“Logical Unit of Work Identifier (LUWId)” on page 210
“GDS Records and Data Mapping” on page 210
“Presentation Services (PS) Header 10” on page 212
“Function Management Header 7 (FMH-7)” on page 213
“Exchange Log Names (XLN) GDS Record” on page 214
“Compare States GDS Record” on page 215
“Conversation Correlator (CC) and Session Id” on page 216

# Appendix D. Appendix D. IBM Communication Server APPC Interface

These programming interfaces from the IBM Communication Server are used by the Component Broker to communicate over SNA. These header files contain the interface information and can be found in the *IBMCS_INSTALL*/SDK/WIN32/H directory, where *IBMCS_INSTALL* is the install directory for IBM Communications Server. For more information, see the IBM manual, *Communication Server Client/Server Communications Programming (SC31-8425)*.

- "IBM Communication Server APPC Interface: Operation Codes" (WINAPPC.H)
- "IBM Communication Server APPC Interface: Verb Parameters" on page 194 (WINPARMS.H)
- "IBM Communication Server APPC Interface: Return Codes" on page 195 (WINRC.H)

## IBM Communication Server APPC Interface: Operation Codes

These operation codes from the IBM Communication Server programming interface are used by the Component Broker to communicate over SNA, which can be found in winappc.h in the *IBMCS_INSTALL*/SDK/WIN32/H directory, where *IBMCS_INSTALL* is the install directory for IBM Communications Server. For more information, see the IBM manual, *Communication Server Client/Server Communications Programming (SC31-8425)*.

```
/*****************************************************************************/
/* Operation codes                                                           */
/*****************************************************************************/
#define AP_TP_STARTED                   0x0014   // See tp_started VCB
#define AP_RECEIVE_ALLOCATE             0x0016
#define AP_TP_ENDED                     0x0013   // See tp_ended VCB
#define AP_B_ALLOCATE                   0x0001   // See allocate VCB
#define AP_B_CONFIRM                    0x0003   // See confirm VCB
#define AP_B_CONFIRMED                  0x0004   // See confirmed VCB
#define AP_B_DEALLOCATE                 0x0005   // See deallocate VCB
#define AP_B_FLUSH                      0x0006
#define AP_B_GET_ATTRIBUTES             0x0007   // See get_attributes VCB
#define AP_B_PREPARE_TO_RECEIVE         0x000A
#define AP_B_RECEIVE_AND_POST           0x000D
#define AP_B_RECEIVE_AND_WAIT           0x000B   // See receive_and_wait VCB
#define AP_B_RECEIVE_EXPEDITED_DATA     0x0009
#define AP_B_RECEIVE_IMMEDIATE          0x000C
#define AP_B_REQUEST_TO_SEND            0x000E
#define AP_B_SEND_CONVERSATION          0x0018
#define AP_B_SEND_DATA                  0x000F   // See send_data VCB
#define AP_B_SEND_ERROR                 0x0010   // See send_error VCB
#define AP_B_SEND_EXPEDITED_DATA        0x0011
#define AP_B_TEST_RTS                   0x0012
#define AP_B_TEST_RTS_AND_POST          0x0081
#define AP_GET_TP_PROPERTIES            0x0017
#define AP_SET_TP_PROPERTIES            0x007E   // See set_tp_properties VCB
#define AP_GET_LU_STATUS                0x007F   // See get_lu_status VCB
#define AP_REGISTER_EVENT               0x2066
#define AP_UNREGISTER_EVENT             0x2067
#define AP_EVENT_INDICATION             0x2068
/*****************************************************************************/
/* Verb parameter values for APPC                                            */
/* (common verb parameter values are in winparms.h)                          */
/*****************************************************************************/
```

```
#define AP_INDEPENDENT_LU               0x00
#define AP_SYSTEM_LIMIT                 0x00
/*****************************************************************************/
/* conversation states                                                       */
/*****************************************************************************/
#define AP_RESET_STATE                  0x00
#define AP_SEND_STATE                   0x01
#define AP_RECEIVE_STATE                0x02
#define AP_CONFIRM_STATE                0x03
#define AP_CONFIRM_SEND_STATE           0x04
#define AP_CONFIRM_DEALL_STATE          0x05
#define AP_PEND_POST_STATE              0x06
#define AP_PEND_DEALL_STATE             0x07
#define AP_END_CONV_STATE               0x08
#define AP_SEND_PENDING_STATE           0x09
/*****************************************************************************/
/* ptr_type                                                                  */
/*****************************************************************************/
#define AP_SYNC_LEVEL                   0x00
#define AP_FLUSH                        0x01
#define AP_P_TO_R_CONFIRM               0x02
/*****************************************************************************/
/* dealloc_type                                                              */
/*****************************************************************************/
#define AP_SYNC_LEVEL                   0x00
#define AP_FLUSH                        0x01
#define AP_CONFIRM_TYPE                 0x0B
#define AP_ABEND                        0x05
#define AP_ABEND_PROG                   0x02
#define AP_ABEND_SVC                    0x03
#define AP_ABEND_TIMER                  0x04
#define AP_TP_NOT_AVAIL_RETRY           0x06
#define AP_TP_NOT_AVAIL_NO_RETRY        0x07
#define AP_TPN_NOT_RECOGNIZED           0x08
#define AP_PIP_DATA_NOT_ALLOWED         0x09
#define AP_PIP_DATA_INCORRECT           0x0A
#define AP_RESOURCE_FAILURE_NO_RETRY    0x0C
#define AP_CONV_TYPE_MISMATCH           0x0D
#define AP_SYNC_LVL_NOT_SUPPORTED       0x0E
#define AP_SECURITY_PARAMS_INVALID      0x0F
#define AP_ATTACH_ERROR                 0xFF
/*****************************************************************************/
/* fill                                                                      */
/*****************************************************************************/
#define AP_BUFFER                       0x00
#define AP_LL                           0x01
/*****************************************************************************/
/* Other flags for the opext field of APPC verbs                             */
/*****************************************************************************/
#define AP_NON_BLOCKING                 0x02
#define AP_OPERATION_INCOMPLETE_FLAG    0x40
/*****************************************************************************/
/* Reserved wait_object values                                               */
/* wait objects are used for non-blocking verbs, supplied in secondary_rc    */
/* field                                                                     */
/*****************************************************************************/
#define AP_BLOCKING_WAIT_OBJECT         0x00000000L
#define AP_NULL_WAIT_OBJECT             0xFFFFFFFFL
/*****************************************************************************/
/* ptr_locks                                                                 */
/*****************************************************************************/
#define AP_SHORT                        0x00
#define AP_LONG                         0x01
/*****************************************************************************/
/* err_type                                                                  */
/*****************************************************************************/
#define AP_PROG                         0x00
```

```
#define AP_SVC                          0x01
#define AP_BACKOUT_NO_RESYNC            0x02
#define AP_BACKOUT_RESYNC               0x03
/*****************************************************************************/
/* rtn_ctl                                                                   */
/*****************************************************************************/
#define AP_WHEN_SESSION_ALLOCATED       0x00
#define AP_IMMEDIATE                    0x01
#define AP_WHEN_EXPEDITED_DATA_RECEIVED 0x00
#define AP_WHEN_SESSION_FREE            0x02
#define AP_WHEN_CONWINNER_ALLOC         0x03
#define AP_WHEN_CONWINNER_ALLOCATED     0x03
#define AP_WHEN_CONV_GROUP_ALLOC        0x04
#define AP_WHEN_CONV_GROUP_ALLOCATED    0x04
#define AP_WHEN_CONLOSER_ALLOC          0x05
/*****************************************************************************/
/* end_type                                                                  */
/*****************************************************************************/
#define AP_SOFT                         0x00
#define AP_HARD                         0x01
#define AP_CANCEL                       0x0A
/*****************************************************************************/
/* data_type                                                                 */
/*****************************************************************************/
#define AP_APPLICATION                  0x00
#define AP_USER_CONTROL_DATA            0x01
#define AP_PS_HEADER                    0x02
#define AP_BASIC_DATA                   0x03
/*****************************************************************************/
/* err_dir                                                                   */
/*****************************************************************************/
#define AP_RCV_DIR_ERROR                0x00
#define AP_SEND_DIR_ERROR               0x01
/*****************************************************************************/
/* what_rcvd                                                                 */
/*****************************************************************************/
#define AP_DATA                         0x0100
#define AP_DATA_COMPLETE                0x0200
#define AP_DATA_INCOMPLETE              0x0400
#define AP_SEND                         0x0001
#define AP_CONFIRM_WHAT_RECEIVED        0x0002
#define AP_CONFIRM_SEND                 0x0003
#define AP_CONFIRM_DEALLOCATE           0x0004
#define AP_USER_CONTROL_DATA_COMPLETE   0x0800
#define AP_USER_CONTROL_DATA_INCMP      0x1000
#define AP_PS_HEADER_COMPLETE           0x2000
#define AP_PS_HEADER_INCOMPLETE         0x4000
#define AP_DATA_SEND                    0x0101
#define AP_DATA_CONFIRM                 0x0102
#define AP_DATA_CONFIRM_SEND            0x0103
#define AP_DATA_CONFIRM_DEALLOCATE      0x0104
#define AP_DATA_COMPLETE_SEND           0x0201
#define AP_DATA_COMPLETE_CONFIRM        0x0202
#define AP_DATA_COMPLETE_CONFIRM_SEND   0x0203
#define AP_DATA_COMPLETE_CONFIRM_DEALL  0x0204
#define AP_UC_DATA_COMPLETE_SEND        0x0801
#define AP_UC_DATA_COMPLETE_CONFIRM     0x0802
#define AP_UC_DATA_COMPLETE_CNFM_SEND   0x0803
#define AP_UC_DATA_COMPLETE_CNFM_DEALL  0x0804
#define AP_PS_HDR_COMPLETE_SEND         0x2001
#define AP_PS_HDR_COMPLETE_CONFIRM      0x2002
#define AP_PS_HDR_COMPLETE_CNFM_SEND    0x2003
#define AP_PS_HDR_COMPLETE_CNFM_DEALL   0x2004
/*****************************************************************************/
/* reason for implied forget indication                                      */
/*****************************************************************************/
#define AP_DATA_FLOW                    0x00
```

```
#define AP_UNBIND                       0x01
#define AP_FAILURE                      0x02
/*****************************************************************************/
/* Opcodes for APPC event indications, used as bit masks, so valid values    */
/* are 1, 2, 4, 8 etc                                                        */
/*****************************************************************************/
#define AP_RECEIVED_DATA                0x0001
/*****************************************************************************/
/* VCB structures                                                            */
/*****************************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: APPC_HDR                                                       */
/*                                                                           */
/* Description: Common APPC vcb header                                       */
/*****************************************************************************/
typedef struct appc_hdr
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
} APPC_HDR;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: TP_STARTED                                                     */
/*                                                                           */
/* Description: TP_STARTED verb control block                                */
/*****************************************************************************/
typedef struct tp_started
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   lu_alias[8];
  unsigned char   tp_id[8];
  unsigned char   tp_name[64];
} TP_STARTED;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: RECEIVE_ALLOCATE                                               */
/*                                                                           */
/* Description: RECEIVE_ALLOCATE verb control block                          */
/*****************************************************************************/
typedef struct receive_allocate
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_name[64];
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   sync_level;
  unsigned char   conv_type;
  unsigned char   user_id[10];
  unsigned char   lu_alias[8];
  unsigned char   plu_alias[8];
  unsigned char   mode_name[8];
  unsigned char   reserv3[2];
  unsigned long   conv_group_id;
  unsigned char   fqplu_name[17];
  unsigned char   pip_incoming;
  unsigned char   conversation_style;
```

```
  unsigned char   reserv4[3];
  unsigned char   password[10];
  unsigned char   reserv5[2];
  unsigned char   dload_id[8];
} RECEIVE_ALLOCATE;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: TP_ENDED                                                           */
/*                                                                               */
/* Description: TP_ENDED verb control block                                      */
/*********************************************************************************/
typedef struct tp_ended
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned char   type;
} TP_ENDED;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: ALLOCATE                                                           */
/*                                                                               */
/* Description: ALLOCATE verb control block                                      */
/*********************************************************************************/
typedef struct allocate
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   conv_type;
  unsigned char   sync_level;
  unsigned char   reserv3[2];
  unsigned char   rtn_ctl;
  unsigned char   conversation_style;
  unsigned long   conv_group_id;
  unsigned long   sense_data;
  unsigned char   plu_alias[8];
  unsigned char   mode_name[8];
  unsigned char   tp_name[64];
  unsigned char   security;
  unsigned char   reserv5[11];
  unsigned char   pwd[10];
  unsigned char   user_id[10];
  unsigned short  pip_dlen;
  unsigned char   *pip_dptr;
  unsigned char   reserv5a;
  unsigned char   fqplu_name[17];
  unsigned char   reserv6[8];
} ALLOCATE;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: CONFIRM                                                            */
/*                                                                               */
/* Description: CONFIRM verb control block                                       */
/*********************************************************************************/
typedef struct confirm
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
```

```
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   rts_rcvd;
  unsigned char   expd_data_rcvd;
} CONFIRM;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: CONFIRMED                                                          */
/*                                                                               */
/* Description: CONFIRMED verb control block                                     */
/**********************************************************************************/
typedef struct confirmed
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
} CONFIRMED;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: DEALLOCATE                                                         */
/*                                                                               */
/* Description: DEALLOCATE verb control block                                    */
/**********************************************************************************/
typedef struct deallocate
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   expd_data_rcvd;
  unsigned char   dealloc_type;
  unsigned short  log_dlen;
  unsigned char   *log_dptr;
  /**********************************************************************************/
  /* If format field set to 1 and syncpoint callback NOT required then             */
  /* these fields must be set to null (0x00)                                       */
  /**********************************************************************************/
  DEALLOCATE_CALLBACK  callback;
  void *          correlator;
  unsigned char   reserv4[4];
} DEALLOCATE;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: FLUSH                                                              */
/*                                                                               */
/* Description: FLUSH verb control block                                         */
/**********************************************************************************/
typedef struct flush
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
} FLUSH;
/**STRUCT-************************************************************************/
```

```
/**STRUCT+*********************************************************************/
/* Structure: GET_ATTRIBUTES                                                  */
/*                                                                            */
/* Description: GET_ATTRIBUTES verb control block                             */
/******************************************************************************/
typedef struct get_attributes
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   reserv3;
  unsigned char   sync_level;
  unsigned char   mode_name[8];
  unsigned char   net_name[8];
  unsigned char   lu_name[8];
  unsigned char   lu_alias[8];
  unsigned char   plu_alias[8];
  unsigned char   plu_un_name[8];
  unsigned char   reserv4[2];
  unsigned char   fqplu_name[17];
  unsigned char   reserv5;
  unsigned char   user_id[10];
  unsigned long   conv_group_id;
  unsigned char   conv_corr_len;
  unsigned char   conv_corr[8];
  unsigned char   reserv6[13];
  unsigned char   luw_id[26];
  unsigned char   sess_id[8];
} GET_ATTRIBUTES;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: PREPARE_TO_RECEIVE                                              */
/*                                                                            */
/* Description: PREPARE_TO_RECEIVE verb control block                         */
/******************************************************************************/
typedef struct prepare_to_receive
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   ptr_type;
  unsigned char   locks;
} PREPARE_TO_RECEIVE;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: RECEIVE_AND_WAIT                                                */
/*                                                                            */
/* Description: RECEIVE_AND_WAIT verb control block                           */
/******************************************************************************/
typedef struct receive_and_wait
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned short  what_rcvd;
```

```
  unsigned char   rtn_status;
  unsigned char   fill;
  unsigned char   rts_rcvd;
  unsigned char   expd_data_rcvd;
  unsigned short  max_len;
  unsigned short  dlen;
  unsigned char   *dptr;
  unsigned char   reserv5[5];
} RECEIVE_AND_WAIT;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: SEND_DATA                                                          */
/*                                                                               */
/* Description: SEND_DATA verb control block                                     */
/**********************************************************************************/
typedef struct send_data
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   rts_rcvd;
  unsigned char   expd_data_rcvd;
  unsigned short  dlen;
  unsigned char   *dptr;
  unsigned char   type;
  unsigned char   reserv4;
} SEND_DATA;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: SEND_ERROR                                                         */
/*                                                                               */
/* Description: SEND_ERROR verb control block                                    */
/**********************************************************************************/
typedef struct send_error
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned long   conv_id;
  unsigned char   rts_rcvd;
  unsigned char   err_type;
  unsigned char   err_dir;
  unsigned char   expd_data_rcvd;
  unsigned short  log_dlen;
  unsigned char   *log_dptr;
} SEND_ERROR;
/**STRUCT-************************************************************************/
/**STRUCT+************************************************************************/
/* Structure: LUW_ID_OVERLAY                                                     */
/*                                                                               */
/* Description: Identifier for Logical Unit of Work                              */
/**********************************************************************************/
typedef struct luw_id_overlay
{
  unsigned char   fqlu_name_len;
  unsigned char   fqlu_name[17];
  unsigned char   instance[6];
  unsigned char   sequence[2];
} LUW_ID_OVERLAY;
/**STRUCT-************************************************************************/
```

```
/**STRUCT+*********************************************************************/
/* Structure: GET_TP_PROPERTIES                                               */
/*                                                                            */
/* Description: GET_TP_PROPERTIES verb control block                          */
/******************************************************************************/
typedef struct get_tp_properties
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned char   tp_name[64];
  unsigned char   lu_alias[8];
  unsigned char   luw_id[26];
  unsigned char   fqlu_name[17];
  unsigned char   reserv3[10];
  unsigned char   user_id[10];
  unsigned char   prot_luw_id[26];
} GET_TP_PROPERTIES;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: GET_LU_STATUS                                                   */
/*                                                                            */
/* Description: GET_LU_STATUS verb control block                              */
/******************************************************************************/
typedef struct get_lu_status
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned char   plu_alias[8];
  unsigned short  active_sess;
  unsigned char   zero_sess;
  unsigned char   reserv3[7];
} GET_LU_STATUS;
/**STRUCT-*********************************************************************/
/**STRUCT+*********************************************************************/
/* Structure: SET_TP_PROPERTIES                                               */
/*                                                                            */
/* Description: SET_TP_PROPERTIES verb control block                          */
/******************************************************************************/
typedef struct set_tp_properties
{
  unsigned short  opcode;
  unsigned char   opext;
  unsigned char   format;
  unsigned short  primary_rc;
  unsigned long   secondary_rc;
  unsigned char   tp_id[8];
  unsigned char   set_prot_id;
  unsigned char   new_prot_id;
  unsigned char   prot_id[26];
  unsigned char   set_unprot_id;
  unsigned char   new_unprot_id;
  unsigned char   unprot_id[26];
  unsigned char   set_user_id;
  unsigned char   set_password;
  unsigned char   user_id[10];
  unsigned char   new_password[10];
} SET_TP_PROPERTIES;
/**STRUCT-*********************************************************************/
/******************************************************************************/
```

```
/* Windows APPC Extension Return Codes                                      */
/***************************************************************************/
#define WAPPCALREADY         0xF000   /* An async call is already outstanding*/
#define WAPPCINVALID         0xF001   /* Async Task Id is invalid            */
#define WAPPCCANCEL          0xF002   /* Blocking call was cancelled         */
#define WAPPCSYSNOTREADY     0xF003   /* Underlying subsystem not started    */
#define WAPPCVERNOTSUPPORTED 0xF004   /* Application version not supported   */
/***************************************************************************/
/* WinAPPCStartup DATA structure                                           */
/***************************************************************************/
#define WAPPCDESCRIPTION_LEN    127
typedef struct tagWAPPCDATA
{
  WORD    wVersion;
  char    szDescription[WAPPCDESCRIPTION_LEN+1];
} WAPPCDATA, * PWAPPCDATA, * LPWAPPCDATA;
/***************************************************************************/
/* API function prototypes                                                 */
/***************************************************************************/
extern void    WINAPI APPC(long);
extern void    WINAPI APPC_C(long);
extern HANDLE  WINAPI WinAsyncAPPC(HWND, long);
extern HANDLE  WINAPI WinAsyncAPPCEx(HANDLE, long);
extern BOOL    WINAPI WinAPPCCleanup(void);
extern BOOL    WINAPI WinAPPCIsBlocking(void);
extern int     WINAPI WinAPPCCancelAsyncRequest( HANDLE );
extern BOOL    WINAPI WinAPPCCancelBlockingCall(void);
extern int     WINAPI WinAPPCStartup(WORD, LPWAPPCDATA);
extern FARPROC WINAPI WinAPPCSetBlockingHook(FARPROC);
extern BOOL    WINAPI WinAPPCUnhookBlockingHook(void);
extern int     WINAPI GetAppcReturnCode(struct appc_hdr *,
                                        UINT,
                                        unsigned char *);
/* winappc.h */
```

**RELATED REFERENCES**

"Appendix D. Appendix D. IBM Communication Server APPC Interface" on page 185

## IBM Communication Server APPC Interface: Verb Parameters

These APPC verb parameters from the IBM Communication Server programming interface are used by the Component Broker to communicate over SNA, which can be found in winparms.h in the *IBMCS_INSTALL*/SDK/WIN32/H directory, where *IBMCS_INSTALL* is the install directory for IBM Communications Server. For more information, see the IBM manual, *Communication Server Client/Server Communications Programming (SC31-8425)*.

```
/***************************************************************************/
/* Verb parameter values                                                   */
/***************************************************************************/
#define AP_NO                           0x00
#define AP_YES                          0x01
#define AP_ONE                          0x00
#define AP_ALL                          0x01
/***************************************************************************/
/*
sync_level                                                              */
/***************************************************************************/
#define AP_NONE                         0x00
#define AP_CONFIRM                      0x01
#define AP_SYNCPT                       0x02
/***************************************************************************/
/* sync_levels_supported                                                   */
```

```
/*       AP_NONE                        0x00                                */
/*       AP_CONFIRM_SYNC_LEVEL          0x01                                */
/*       AP_EITHER                      0x02        defined elsewhere.      */
/*****************************************************************************/
/*****************************************************************************/
/* opext and
conv_type                                                          */
/*****************************************************************************/
#define AP_BASIC_CONVERSATION          0x00
#define AP_MAPPED_CONVERSATION         0x01
/*****************************************************************************/
/* Flags to use in APPC Queue-Level Non-Blocking Mode                        */
/*****************************************************************************/
#define AP_NON_BLOCKING                0x02
#define AP_OPERATION_INCOMPLETE_FLAG   0x40
/*****************************************************************************/
/*
"security"                                                           */
/*****************************************************************************/
#define AP_NONE                        0x00
#define AP_SAME                        0x01
#define AP_PGM                         0x02
/*****************************************************************************/
/*
send_type                                                          */
/*****************************************************************************/
#define AP_NONE                        0x00
#define AP_SEND_DATA_FLUSH             0x01
#define AP_SEND_DATA_CONFIRM           0x02
#define AP_SEND_DATA_P_TO_R_FLUSH      0x03
#define AP_SEND_DATA_P_TO_R_SYNC_LEVEL 0x04
#define AP_SEND_DATA_P_TO_R_CONFIRM    0x08
#define AP_SEND_DATA_DEALLOC_FLUSH     0x05
#define AP_SEND_DATA_DEALLOC_SYNC_LEVEL 0x06
#define AP_SEND_DATA_DEALLOC_CONFIRM   0x09
#define AP_SEND_DATA_DEALLOC_ABEND     0x07
/*****************************************************************************/
/* Constants for security_details field in PLU_DETAIL structure.  These      */
/* constants have the same values as the equivalent constants for the bit    */
/* settings for BIND byte 23                                                 */
/*****************************************************************************/
#define AP_CONVERSATION_LEVEL_SECURITY (unsigned char)0x10
#define AP_PERSISTENT_VERIFICATION     (unsigned char)0x01
#define AP_ALREADY_VERIFIED            (unsigned char)0x02
#define AP_PASSWORD_SUBSTITUTION       (unsigned char)0x04
/* winparms.h */
```

**RELATED REFERENCES**

“Appendix D. Appendix D. IBM Communication Server APPC Interface” on page 185

## IBM Communication Server APPC Interface: Return Codes

These return codes from the IBM Communication Server programming interface are used by the Component Broker to communicate over SNA, which can be found in winrc.h in the *IBMCS_INSTALL*/SDK/WIN32/H directory, where *IBMCS_INSTALL* is the install directory for IBM Communications Server. This header file contains the primary and secondary return codes. For more information, see the IBM manual, *Communication Server Client/Server Communications Programming (SC31-8425)*.

```
/*****************************************************************************/
/*
Primary return codes                                                 */
/*****************************************************************************/
```

```
#define AP_OK                                   0x0000
#define AP_ACTIVATION_FAIL_NO_RETRY             0x0410
#define AP_ACTIVATION_FAIL_RETRY                0x0310
#define AP_ALLOCATION_ERROR                     0x0300
#define AP_BACKED_OUT                           0x2200
#define AP_BUFFER_PROVIDED_TOO_SMALL            0x4500
#define AP_BUFFER_TOO_SMALL                     AP_BUFFER_PROVIDED_TOO_SMALL
#define AP_CANCELLED                            0x2100
#define AP_CNOS_LOCAL_RACE_REJECT               0x1700
#define AP_CNOS_PARTNER_LU_REJECT               0x1800
#define AP_COMM_SUBSYSTEM_ABENDED               0x03F0
#define AP_COMM_SUBSYSTEM_NOT_LOADED            0x04F0
#define AP_CONVERSATION_TYPE_MIXED              0x1900
#define AP_CONVERSATION_ENDED                   0x4200
#define AP_CONV_BUSY                            0x05F0
#define AP_CONV_FAILURE_NO_RETRY                0x1000
#define AP_CONV_FAILURE_RETRY                   0x0F00
#define AP_DEALLOC_ABEND                        0x0500
#define AP_DEALLOC_ABEND_PROG                   0x0600
#define AP_DEALLOC_ABEND_SVC                    0x0700
#define AP_DEALLOC_ABEND_TIMER                  0x0800
#define AP_DEALLOC_NORMAL                       0x0900
#define AP_BACKED_OUT                           0x2300
#define AP_ERROR_INDICATION                     0x4300
#define AP_EXPD_NOT_SUPPORTED_BY_LU             0x4400
#define AP_EXP_DATA_NOT_SUPPORTED_BY_LU         AP_EXPD_NOT_SUPPORTED_BY_LU
#define AP_FUNCTION_NOT_SUPPORTED               0x0610
#define AP_INDICATION                           0x0210
#define AP_INVALID_KEY                          0x20F0
#define AP_INVALID_VERB_SEGMENT                 0x08F0
#define AP_LS_FAILURE                           0x2300
#define AP_LU_SESS_LIMIT_EXCEEDED               0x0510
#define AP_NODE_NOT_STARTED                     0x1B00
#define AP_NODE_STOPPING                        0x1A00
#define AP_OPERATION_INCOMPLETE                 0x4000
#define AP_OPERATION_NOT_ACCEPTED               0x4100
#define AP_PARAMETER_CHECK                      0x0100
#define AP_PATH_SWITCH_DISABLED                 0x0710
#define AP_PROG_ERROR_NO_TRUNC                  0x0C00
#define AP_PROG_ERROR_PURGING                   0x0E00
#define AP_PROG_ERROR_TRUNC                     0x0D00
#define AP_REPLACED                             0x0080
#define AP_STACK_TOO_SMALL                      0x15F0
#define AP_STATE_CHECK                          0x0200
#define AP_SVC_ERROR_NO_TRUNC                   0x1100
#define AP_SVC_ERROR_PURGING                    0x1300
#define AP_SVC_ERROR_TRUNC                      0x1200
#define AP_THREAD_BLOCKING                      0x06F0
#define AP_TP_BUSY                              0x02F0
#define AP_UNEXPECTED_DOS_ERROR                 0x11F0
#define AP_UNEXPECTED_SYSTEM_ERROR              0x11F0
#define AP_UNSUCCESSFUL                         0x1400
#define AP_ACCESS_DENIED                        0x0101
#define AP_INVALID_VERB                         0xFFFF
/*****************************************************************************/
/*
Secondary return codes                                                     */
/*****************************************************************************/
#define AP_ACTIVATION_LIMITS_REACHED            0x3E100000L
#define AP_ALLOCATE_NOT_PENDING                 0x09050000L
#define AP_ALLOCATION_ERROR_PENDING             0x00000300L
#define AP_ALLOCATION_FAILURE_NO_RETRY          0x04000000L
#define AP_ALLOCATION_FAILURE_RETRY             0x05000000L
#define AP_ALL_APPL_ALREADY_REGISTERED          0x48080000L
#define AP_ALL_MODE_MUST_RESET                  0x53010000L
#define AP_ALREADY_STARTING                     0xC0010000L
#define AP_APPL_ALREADY_REGISTERED              0x63080000L
```

```
#define AP_APPL_NOT_REGISTERED                  0x4A080000L
#define AP_AS_NEGOTIATED                        0x07000000L
#define AP_AS_SPECIFIED                         0x00000000L
#define AP_ATTACH_MANAGER_INACTIVE              0x08050000L
#define AP_AUTOACT_EXCEEDS_SESSLIM              0x52010000L
#define AP_BAD_CONV_ID                          0x02000000L
#define AP_BAD_CONV_TYPE                        0x11000000L
#define AP_BAD_DUPLEX_TYPE                      0x22000000L
#define AP_BAD_CONV_STYLE    AP_BAD_DUPLEX_TYPE
#define AP_BAD_ERROR_DIRECTION                  0x05010000L
#define AP_BAD_DLOAD_ID                         0x03000001L
#define AP_BAD_LL                               0xF1000000L
#define AP_BAD_LU_ALIAS                         0x03000000L
#define AP_BAD_MODE_NAME                        0x57010000L
#define AP_BAD_PARTNER_LU_ALIAS                 0x5B010000L
#define AP_BAD_REMOTE_LU_ALIAS                  0x03000002L
#define AP_BAD_RETURN_CONTROL                   0x14000000L
#define AP_BAD_RETURN_STATUS_WITH_DATA          0xD7000000L
#define AP_BAD_SECURITY                         0x13000000L
#define AP_BAD_SNASVCMG_LIMITS                  0x54010000L
#define AP_BAD_SYNC_LEVEL                       0x12000000L
#define AP_BAD_TP_ID                            0x01000000L
#define AP_BAD_TYPE                             0x50020000L
#define AP_BO_NO_RESYNC                         0x00002408L
#define AP_BO_RESYNC                            0x01002408L
#define AP_CANT_CHANGE_TO_ZERO                  0x5D010000L
#define AP_CANT_DELETE_ADJ_ENDNODE              0x5C100000L
#define AP_CANT_DELETE_CP_LU                    0xB7070000L
#define AP_CANT_MODIFY_PORT_NAME                0x04100000L
#define AP_CANT_RAISE_LIMITS                    0x51010000L
#define AP_CHANGE_SRC_DRAINS                    0x5D010000L
#define AP_CLASHING_NAU_RANGE                   0x1A900000L
#define AP_CNOS_ACCEPTED                        0x00000000L
#define AP_CNOS_COMMAND_RACE_REJECT             0x5F010000L
#define AP_CNOS_IMPLICIT_PARALLEL               0x50010000L
#define AP_CNOS_MODE_CLOSED                     0x56010000L
#define AP_CNOS_MODE_NAME_REJECT                0x57010000L
#define AP_CNOS_NEGOTIATED                      0x07000000L
#define AP_CNOS_REJECT                          0x2C100000L
#define AP_CONFIRM_BAD_STATE                    0x32000000L
#define AP_CONFIRMED_BAD_STATE                  0x41000000L
#define AP_CONFIRM_INVALID_FOR_FDX              0x34000000L
#define AP_CONFIRM_NOT_LL_BDY                   0x33000000L
#define AP_CONFIRM_ON_SYNC_LEVEL_NONE           0x31000000L
#define AP_CONFIRMED_BAD_STATE                  0x41000000L
#define AP_CONFIRMED_INVALID_FOR_FDX            0x42000000L
#define AP_CONVERSATION_TYPE_MISMATCH           0x34600810L
#define AP_COS_NAME_NOT_DEFD                    0x10080000L
#define AP_COS_TABLE_FULL                       0x64100000L
#define AP_CPSVCMG_ALREADY_DEFD                 0x21020000L
#define AP_CPSVCMG_MODE_NOT_ALLOWED             0x19050000L
#define AP_CP_OR_SNA_SVCMG_UNDELETABLE          0xF3010000L
#define AP_DEACT_CG_INVALID_CGID                0x6C020000L
#define AP_DEALLOC_ABEND_PROG_PENDING           0x00000600L
#define AP_DEALLOC_ABEND_SVC_PENDING            0x00000700L
#define AP_DEALLOC_ABEND_TIMER_PENDING          0x00000800L
#define AP_DEALLOC_BAD_TYPE                     0x51000000L
#define AP_DEALLOC_CONFIRM_BAD_STATE            0x53000000L
#define AP_DEALLOC_FLUSH_BAD_STATE              0x52000000L
#define AP_DEALLOC_LOG_LL_WRONG                 0x57000000L
#define AP_DEALLOC_NOT_LL_BDY                   0x55000000L
#define AP_DEF_LINK_INVALID_SECURITY            0x22080000L
#define AP_DEF_PLU_INVALID_FQ_NAME              0x74020000L
#define AP_DEL_MODE_DEFAULT_SPCD                0xF4010000L
#define AP_DEPENDENT_LU_NOT_SUPPORTED           0x0F900000L
#define AP_DIR_ENTRY_PARENT                     0x38100000L
#define AP_DISPLAY_INFO_EXCEEDS_LEN             0xB4010000L
```

```
#define AP_DISPLAY_INVALID_CONSTANT             0xB5010000L
#define AP_DLC_ACTIVE                           0x01100000L
#define AP_DLC_DEACTIVATING                     0x86020000L
#define AP_DLC_INACTIVE                         0x40100000L
#define AP_DLUR_NOT_SUPPORTED                   0x5F100000L
#define AP_DLUS_CAPS_MISMATCH                   0x08900000L
#define AP_DLUS_REJECTED                        0x07900000L
#define AP_DSPU_ALREADY_DEFINED                 0x13900000L
#define AP_DSPU_SERVICES_NOT_SUPPORTED          0x11900000L
#define AP_DUPLICATE                            0x8D020000L
#define AP_DUPLICATE_ADJ_NODE_ID                0x04100000L
#define AP_DUPLICATE_CP_NAME                    0x02100000L
#define AP_DUPLICATE_DEST_ADDR                  0x03100000L
#define AP_DUPLICATE_PORT                       0x10100000L
#define AP_DUPLICATE_PORT_NAME                  0x06100000L
#define AP_DUPLICATE_PORT_NUMBER                0x05100000L
#define AP_DUPLICATE_TG_NUMBER                  0x15530000L
#define AP_DYNAMIC_LOAD_ALREADY_REGD            0x46100000L
#define AP_ERROR                                0x13400000L
#define AP_EXCEEDS_MAX_ALLOWED                  0x5C010000L
#define AP_EXPD_BAD_RETURN_CONTROL              0x26010000L
#define AP_EXPD_DATA_BAD_CONV_STATE             0x27010000L
#define AP_FDX_NOT_SUPPORTED_BY_LU              0x23000000L
#define AP_FLUSH_NOT_SEND_STATE                 0x61000000L
#define AP_FORCED                               0xB7020000L
#define AP_HPR_NOT_SUPPORTED                    0x62100000L
#define AP_IMPLICIT_REQUEST_FAILED              0x6A080000L
#define AP_IMPLICIT_REQUEST_REJECTED            0x65080000L
#define AP_INVALID_ACTIVE_TRANSACTION           0x66080000L
#define AP_INVALID_ADJ_NNCP_NAME                0x4A100000L
#define AP_INVALID_APPLICATION_NAME             0x61080000L
#define AP_INVALID_AUTO_ACT_SUPP                0xB5020000L
#define AP_INVALID_BACK_LEVEL_SUPPORT           0x15000000L
#define AP_INVALID_BKUP_DLUS_NAME               0x15900000L
#define AP_INVALID_BTU_SIZE                     0x44100000L
#define AP_INVALID_BYTE_COST                    0xD1010000L
#define AP_INVALID_CATEGORY_NAME                0x67080000L
#define AP_INVALID_CLEANUP_TYPE                 0x24100000L
#define AP_INVALID_CNOS_SLIM                    0x17020000L
#define AP_INVALID_CN_NAME                      0x21080000L
#define AP_INVALID_CONV_TYPE                    0xA1020000L
#define AP_INVALID_CONVERSATION_TYPE  AP_INVALID_CONV_TYPE
#define AP_INVALID_COS_NAME                     0x25100000L
#define AP_INVALID_COS_SNASVCMG_MODE            0x1C020000L
#define AP_INVALID_CP_NAME                      0xCA010000L
#define AP_INVALID_DATA_SEGMENT                 0x06000000L
#define AP_INVALID_DATA_SIZE                    0x62080000L
#define AP_INVALID_DATA_TYPE                    0x05070000L
#define AP_INVALID_DAYS_LEFT                    0x65100000L
#define AP_ANYNET_NOT_SUPPORTED                 0x66100000L
#define AP_INVALID_DEFAULT_RU_SIZE              0x1D020000L
#define AP_INVALID_DESTINATION                  0x68080000L
#define AP_INVALID_DISCOVERY_SUPPORT            0x67100000L
#define AP_INVALID_DLC                          0x10050000L
#define AP_INVALID_DLC_NAME                     0x07100000L
#define AP_INVALID_DLC_TYPE                     0x08100000L
#define AP_INVALID_DLUS_NAME                    0x00900000L
#define AP_INVALID_DNST_LU_NAME                 0x53100000L
#define AP_INVALID_DRAIN                        0x27100000L
#define AP_INVALID_DRAIN_SOURCE                 0x20100000L
#define AP_INVALID_DRAIN_TARGET                 0x21100000L
#define AP_INVALID_DSPU_NAME                    0x12900000L
#define AP_INVALID_DSPU_SERVICES                0x10900000L
#define AP_INVALID_DUPLEX_SUPPORT               0x17900000L
#define AP_INVALID_DYNAMIC_LOAD                 0x24600810L
#define AP_INVALID_EFFECTIVE_CAPACITY           0x24080000L
#define AP_INVALID_ENABLE_POOL                  0x50300000L
```

```
#define AP_INVALID_ENABLED                      0x25600810L
#define AP_INVALID_FILTER_OPTION                0x0C900000L
#define AP_INVALID_FORCE                        0x22100000L
#define AP_INVALID_FP_NAME                      0x64080000L
#define AP_INVALID_FQPCID                       0x4E100000L
#define AP_INVALID_FQ_LU_NAME                   0xFD010000L
#define AP_INVALID_FQ_OWNING_CP_NAME            0xDB020000L
#define AP_INVALID_HOST_LU_NAME                 0x54100000L
#define AP_INVALID_ISR_THRESHOLDS               0x5A100000L
#define AP_INVALID_LENGTH                       0x59100000L
#define AP_INVALID_LIMITED_RESOURCE             0xCE010000L
#define AP_INVALID_LINK_ACTIVE_LIMIT            0x09100000L
#define AP_INVALID_LINK_ENABLE                  0xBA020000L
#define AP_INVALID_LINK_NAME                    0xC1010000L
#define AP_INVALID_LINK_NAME_SPECIFIED          0xB0020000L
#define AP_INVALID_LIST_OPTION                  0x47100000L
#define AP_INVALID_LS_ROLE                      0x43100000L
#define AP_INVALID_LU_ALIAS                     0xB1020000L
#define AP_INVALID_LU_NAME                      0x29100000L
#define AP_INVALID_MAX_IFRM_RCVD                0x57100000L
#define AP_INVALID_MAX_NEGOT_SESS_LIM           0x14020000L
#define AP_INVALID_MAX_RU_SIZE_UPPER            0x19020000L
#define AP_INVALID_MDS_MU_FORMAT                0x46080000L
#define AP_INVALID_MIN_CONWINNERS               0x1E020000L
#define AP_INVALID_MODE_NAME                    0x15020000L
#define AP_INVALID_MODE_NAME_SELECT             0x1E100000L
#define AP_INVALID_MS_CATEGORY                  0x60080000L
#define AP_INVALID_NAME_LEN                     0xC5020000L
#define AP_INVALID_NAU_ADDRESS                  0x50100000L
#define AP_INVALID_NAU_RANGE                    0x1B900000L
#define AP_INVALID_NETID_LEN                    0xC6020000L
#define AP_INVALID_NEW_PROT                     0x01070000L
#define AP_INVALID_NEW_UNPROT                   0x03070000L
#define AP_INVALID_NODE                         0x4B100000L
#define AP_INVALID_NODE_TYPE                    0xC4020000L
#define AP_INVALID_NODE_TYPE_FOR_HPR            0xC8020000L
#define AP_INVALID_NUMBER_OF_NODE_ROWS          0x02080000L
#define AP_INVALID_NUMBER_OF_TG_ROWS            0x09080000L
#define AP_INVALID_NUM_DSLU_TEMPLATES           0x1C900000L
#define AP_INVALID_NUM_LS_SPECIFIED             0xB2020000L
#define AP_INVALID_NUM_LUS                      0x5B100000L
#define AP_INVALID_NUM_PORTS_SPECIFIED          0x0B100000L
#define AP_INVALID_OP_CODE                      0x2D100000L
#define AP_INVALID_ORIGIN_CP_NAME               0x47080000L
#define AP_INVALID_ORIGIN_NODE                  0x4C100000L
#define AP_INVALID_PASSWORD                     0x91020000L
#define AP_INVALID_PIP_ALLOWED                  0x26600810L
#define AP_INVALID_PLU_NAME                     0x1C100000L
#define AP_INVALID_POOL_NAME                    0x4F100000L
#define AP_INVALID_PORT_NAME                    0x0C100000L
#define AP_INVALID_PORT_TYPE                    0x0D100000L
#define AP_INVALID_PRIORITY                     0x52100000L
#define AP_INVALID_PRLL_SESS_SUPP               0x28100000L
#define AP_INVALID_PROCESS                      0x25050000L
#define AP_INVALID_PROFILE                      0x93020000L
#define AP_INVALID_PROPAGATION_DELAY            0x23080000L
#define AP_INVALID_PU_ID                        0x02900000L
#define AP_INVALID_PU_NAME                      0x56100000L
#define AP_INVALID_PU_TYPE                      0x56600000L
#define AP_INVALID_RECV_PACING_WINDOW           0x16020000L
#define AP_INVALID_RESOURCE_TYPE                0x5D100000L
#define AP_INVALID_RESPONSIBLE                  0x1F100000L
#define AP_INVALID_RES_NAME                     0x48100000L
#define AP_INVALID_RES_TYPE                     0x49100000L
#define AP_INVALID_RTP_CONNECTION               0x60100000L
#define AP_INVALID_SEMAPHORE_HANDLE             0xD6000000L
#define AP_INVALID_SESSION_ID                   0x12050000L
```

```
#define AP_INVALID_SESSION_LIMIT               0x26100000L
#define AP_INVALID_SET_NEGOTIABLE              0x1D100000L
#define AP_INVALID_SET_PROT                    0x00070000L
#define AP_INVALID_SET_UNPROT                  0x02070000L
#define AP_INVALID_SET_USER                    0x04070000L
#define AP_INVALID_SET_PASSWORD                0x09070000L
#define AP_INVALID_SNASVCMG_MODE_LIMIT         0x1A020000L
#define AP_INVALID_SOLICIT_SSCP_SESS           0x14900000L
#define AP_INVALID_STATS_TYPE                  0x06070000L
#define AP_INVALID_STOP_TYPE                   0x0D900000L
#define AP_INVALID_SYM_DEST_NAME               0x58100000L
#define AP_INVALID_SYNC_LEVEL                  0xA3020000L
#define AP_INVALID_TABLE_TYPE                  0x07070000L
#define AP_INVALID_TARGET_PACING_CNT           0x18020000L
#define AP_INVALID_TEMPLATE_NAME               0x19900000L
#define AP_INVALID_TG                          0x4D100000L
#define AP_INVALID_TG_CHARS                    0x18030000L
#define AP_INVALID_TG_NUMBER                   0x15500000L
#define AP_INVALID_TIME_COST                   0xD6010000L
#define AP_INVALID_TP_NAME                     0xA0020000L
#define AP_INVALID_TYPE                        0x69080000L
#define AP_INVALID_UNINT_PLU_NAME              0x7C020000L
#define AP_INVALID_UPDATE_TYPE                 0x37100000L
#define AP_INVALID_USERID                      0x90020000L
#define AP_INVALID_USER_DEF_1                  0xC3010000L
#define AP_INVALID_USER_DEF_2                  0xC4010000L
#define AP_INVALID_USER_DEF_3                  0xC5010000L
#define AP_INVALID_WILDCARD_NAME               0x8C020000L
#define AP_KEY_APPL_ALREADY_REGISTERED         0x42080000L
#define AP_LAST_LINK_ON_ACTIVE_PORT            0x45100000L
#define AP_LINK_ACT_BY_LOCAL                   0x15100000L
#define AP_LINK_ACT_BY_REMOTE                  0x14100000L
#define AP_LINK_DEACTIVATED                    0x13100000L
#define AP_LINK_DEACT_IN_PROGRESS              0x12100000L
#define AP_LINK_NOT_DEFD                       0x17100000L
#define AP_LOCAL_CP_NAME                       0xD7010000L
#define AP_LS_ACTIVE                           0xDA010000L
#define AP_LU_ALIAS_ALREADY_USED               0xB9020000L
#define AP_LU_ALIAS_CANT_BE_CHANGED            0xB8020000L
#define AP_LU_ALREADY_DEFINED                  0x3B100000L
#define AP_IMPLICIT_LU_DEFINED                 0x3C100000L
#define AP_LU_DETACHED                         0x5E010000L
#define AP_LU_NAME_POOL_NAME_CLASH             0x51100000L
#define AP_LU_NAME_WILDCARD_NAME_CLASH         0x8E020000L
#define AP_LU_NAU_ADDR_ALREADY_DEFD            0x12020000L
#define AP_MIN_GT_TOTAL                        0x55010000L
#define AP_MISSING_CP_NAME                     0x15510000L
#define AP_MISSING_CP_TYPE                     0x15520000L
#define AP_MISSING_TG_NUMBER                   0x15550000L
#define AP_MODE_ACTIVE                         0x1A100000L
#define AP_MODE_CLOSED                         0x56010000L
#define AP_MODE_NAME_NOT_DEFD                  0xF5010000L
#define AP_MODE_NOT_RESET                      0x2A100000L
#define AP_MODE_RESET                          0x2B100000L
#define AP_MODE_SESS_LIM_EXCEEDS_NEG           0x20020000L
#define AP_MODE_UNDELETABLE                    0xF6010000L
#define AP_MS_APPL_NAME_ALREADY_REGD           0x40080000L
#define AP_MS_APPL_NAME_NOT_REGD               0x44080000L
#define AP_NODE_ALREADY_STARTED                0x39100000L
#define AP_NODE_FAILED_TO_START                0x3A100000L
#define AP_NODE_ROW_WGT_LESS_THAN_LAST         0x04080000L
#define AP_NO_DEFAULT_DLUS_DEFINED             0x01900000L
#define AP_NO_LINKS_DEFINED                    0x41100000L
#define AP_NO_PORTS_DEFINED_ON_DLC             0x0F100000L
#define AP_NO_PROFILES                         0x33100000L
#define AP_NO_USE_OF_SNASVCMG                  0x17000000L
#define AP_NO_USE_OF_SNASVCMG_CPSVCMG          0x17000000L
```

```
#define AP_P_TO_R_INVALID_FOR_FDX              0xA5000000L
#define AP_PARALLEL_TGS_NOT_ALLOWED            0x15570000L
#define AP_PARALLEL_TGS_NOT_SUPPORTED          0x3F100000L
#define AP_PARTNER_NOT_FOUND                   0x13200000L
#define AP_PARTNER_NOT_RESPONDING              0x13300000L
#define AP_PATH_SWITCH_IN_PROGRESS             0x61100000L
#define AP_PIP_LEN_INCORRECT                   0x16000000L
#define AP_PIP_NOT_ALLOWED                     0x31600810L
#define AP_PIP_NOT_SPECIFIED_CORRECTLY         0x32600810L
#define AP_PLU_ACTIVE                          0x1B100000L
#define AP_PLU_ALIAS_ALREADY_USED              0xB4020000L
#define AP_CANT_DELETE_IMPLICIT_LU             0xB6020000L
#define AP_PLU_ALIAS_CANT_BE_CHANGED           0xB3020000L
#define AP_PORT_ACTIVE                         0x0E100000L
#define AP_PORT_DEACTIVATED                    0x08070000L
#define AP_PORT_INACTIVE                       0x3D100000L
#define AP_PS_CREATION_FAILURE                 0x18100000L
#define AP_PU_ALREADY_ACTIVATING               0x03900000L
#define AP_PU_ALREADY_ACTIVE                   0x05900000L
#define AP_PU_ALREADY_DEACTIVATING             0x04900000L
#define AP_PU_ALREADY_DEFINED                  0x0E900000L
#define AP_PU_CONC_NOT_SUPPORTED               0x5E100000L
#define AP_PU_FAILED_ACTPU                     0x09900000L
#define AP_PU_NOT_ACTIVE                       0x06900000L
#define AP_PU_NOT_DEFINED                      0x55100000L
#define AP_PU_NOT_RESET                        0x0A900000L
#define AP_PU_OWNS_LUS                         0x0B900000L
#define AP_PW_SUB_NOT_SUPP_ON_SESS             0x26050000L
#define AP_P_TO_R_INVALID_TYPE                 0xA1000000L
#define AP_P_TO_R_NOT_LL_BDY                   0xA2000000L
#define AP_P_TO_R_NOT_SEND_STATE               0xA3000000L
#define AP_QUEUE_PROHIBITED                    0x18900000L
#define AP_RCV_AND_POST_BAD_FILL               0xD5000000L
#define AP_RCV_AND_POST_BAD_STATE              0xD1000000L
#define AP_RCV_AND_POST_NOT_LL_BDY             0xD2000000L
#define AP_RCV_AND_WAIT_BAD_FILL               0xB5000000L
#define AP_RCV_AND_WAIT_BAD_STATE              0xB1000000L
#define AP_RCV_AND_WAIT_NOT_LL_BDY             0xB2000000L
#define AP_RCV_EXPD_INVALID_LENGTH             0x25010000L
#define AP_RECEIVE_EXPD_INVALID_LENGTH         AP_RCV_EXPD_INVALID_LENGTH
#define AP_RCV_IMMD_BAD_FILL                   0xC4000000L
#define AP_RCV_IMMD_BAD_STATE                  0xC1000000L
#define AP_RESET_SNA_DRAINS                    0x59010000L
#define AP_RESOURCE_NAME_NOT_ALLOWED           0xB70B0000L
#define AP_RTP_NOT_SUPPORTED                   0x63100000L
#define AP_R_T_S_BAD_STATE                     0xE1000000L
#define AP_R_T_S_INVALID_FOR_FDX               0xE2000000L
#define AP_SECURITY_NOT_VALID                  0x51600F08L
#define AP_SEC_REQUESTED_NOT_SUPPORTED         0x16900000L
#define AP_SEND_DATA_CONFIRM_ON_SYNC_LEVEL_NONE 0xF5000000L
#define AP_SEND_DATA_CONFIRM_SYNC_NONE         0xF5000000L
#define AP_SEND_DATA_INVALID_TYPE              0xF4000000L
#define AP_SEND_DATA_NOT_LL_BDY                0xF6000000L
#define AP_SEND_DATA_NOT_SEND_STATE            0xF2000000L
#define AP_SEND_ERROR_BAD_STATE                0x04010000L
#define AP_SEND_ERROR_BAD_TYPE                 0x03010000L
#define AP_SEND_ERROR_LOG_LL_WRONG             0x02010000L
#define AP_SEND_EXPD_INVALID_LENGTH            0x24010000L
#define AP_SEND_TYPE_INVALID_FOR_FDX           0xF7000000L
#define AP_SINGLE_NOT_SRC_RESP                 0x5A010000L
#define AP_SNASVCMG_RESET_NOT_ALLOWED          0x67010000L
#define AP_SNA_DEFD_COS_CANT_BE_CHANGE         0x0A080000L
#define AP_SNA_DEFD_COS_CANT_BE_DELETE         0x11080000L
#define AP_SPCF_APPL_ALREADY_REGD              0x45080000L
#define AP_SSCP_PU_SESSION_NOT_ACTIVE          0x01030000L
#define AP_STOP_DLC_PENDING                    0x42100000L
#define AP_STOP_PORT_PENDING                   0x11100000L
```

```
#define AP_SYNC_LEVEL_NOT_SUPPORTED             0x41600810L
#define AP_SYSTEM_TP_CANT_BE_CHANGED            0x22600810L
#define AP_SYSTEM_TP_CANT_BE_DELETED            0x23600810L
#define AP_TEST_INVALID_FOR_FDX                 0x23010000L
#define AP_TG_NUMBER_IN_USE                     0x15540000L
#define AP_TG_ROW_WGT_LESS_THAN_LAST            0x05080000L
#define AP_TOO_MANY_PROFILES                    0x36100000L
#define AP_TOO_MANY_TPS                         0x43020000L
#define AP_TP_ACTIVE                            0x19100000L
#define AP_TP_NAME_NOT_RECOGNIZED               0x21600810L
#define AP_TRANS_PGM_NOT_AVAIL_NO_RETRY         0x00004c08L
#define AP_TRANS_PGM_NOT_AVAIL_NO_RTRY          0x00004C08L
#define AP_TRANS_PGM_NOT_AVAIL_RETRY            0x31604B08L
#define AP_UNDEFINED_TP_NAME                    0x06050000L
#define AP_UNKNOWN_ERROR_TYPE_PENDING           0x00001100L
#define AP_UNKNOWN_PARTNER_MODE                 0x18000000L
#define AP_UNKNOWN_USER                         0x32100000L
#define AP_UNRECOGNIZED_DEACT_TYPE              0x0E050000L
#define SV_DATA_EXCEEDS_RU_SIZE                 0x02030000L
#define SV_INVALID_DATA_SIZE                    0x62080000L
#define SV_INVALID_DATA_TYPE                    0x03030000L
#define SV_SSCP_PU_SESSION_NOT_ACTIVE           0x01030000L
#define AP_MEMORY_SHORTAGE                      0x00000001L
#define AP_DIRECTORY_FULL                       0x00000007L
#define AP_INVALID_ALERT_NUMBER                 0x00005521L
#define AP_INVALID_RTM_THRESHOLD                0x00005523L
#define AP_INVALID_RTM_TIMER_OPTION             0x00005524L
#define AP_INVALID_EMULATOR_USER                0x00005536L
#define AP_INVALID_SESSION_NAME                 0x00005537L
#define AP_INVALID_SESSION_TYPE                 0x00005538L
#define AP_SECURITY_NOT_VALID_PASSWORD_EXPIRED                      0x00FF0F08L
#define AP_SECURITY_NOT_VALID_PASSWORD_INVALID                      0x01FF0F08L
#define AP_SECURITY_NOT_VALID_USERID_REVOKED                        0x02FF0F08L
#define AP_SECURITY_NOT_VALID_USERID_INVALID                        0x03FF0F08L
#define AP_SECURITY_NOT_VALID_USERID_MISSING                        0x04FF0F08L
#define AP_SECURITY_NOT_VALID_PASSWORD_MISSING                      0x05FF0F08L
#define AP_SECURITY_NOT_VALID_GROUP_INVALID                         0x06FF0F08L
#define AP_SECURITY_NOT_VALID_USERID_REVOKED_IN_GROUP               0x07FF0F08L
#define AP_SECURITY_NOT_VALID_USERID_NOT_DEFD_TO_GROUP              0x08FF0F08L
#define AP_SECURITY_NOT_VALID_NOT_AUTHORIZED_AT_REMOTE_LU           0x09FF0F08L
#define AP_SECURITY_NOT_VALID_NOT_AUTHORIZED_FROM_LOCAL_LU          0x0AFF0F08L
#define AP_SECURITY_NOT_VALID_NOT_AUTHORIZED_TO_TRANSACTION_PROGRAM 0x0BFF0F08L
#define AP_SECURITY_NOT_VALID_INSTALLATION_EXIT_FAILED              0x0CFF0F08L
#define AP_SECURITY_NOT_VALID_PROCESSING_FAILURE                    0x0DFF0F08L
#define AP_SECURITY_NOT_VALID_PROTOCOL_VIOLATION                    0x0EFF0F08L
/* winrc.h */
```

**RELATED REFERENCES**

"Appendix D. Appendix D. IBM Communication Server APPC Interface" on page 185

# Appendix E. Transaction Service Exceptions

The following explains the exceptions that are listed with Transaction Service messages in the activity log:

- **CORBA::BAD_INV_ORDER**
  This exception is returned when an internal method is called out of turn. This is usually an internal logic error in Component Broker. A trace of the Transaction Service during the failing request is required to determine the cause of this problem.
- **CORBA::BAD_PARAM**
  This exception is returned when values supplied on the APPC Connection are not acceptable SNA values. The minor code raised with the exception indicates which field in the APPC Connection has caused the exception to be raised and why.
- **CORBA::COMM_FAILURE**
  This exception is raised when an SNA communication failure is detected. This can be caused by an SNA network failure, shutdown of the tier-3 system or stopping the IBM Communication Server node locally.
- **CORBA::INITIALIZE**
  This exception is returned from a CosTransactions::Current::begin() method call if the Transaction Service did not initialize successfully in the server. This may be, for example, because the Transaction Service Log files are not available. The activity log messages produced by the Transaction Service when the server starts indicates why transactions are not available.
- **CORBA::INTERNAL**
  This exception is raised if the transaction service detects an internal logic error. A trace of the transaction service during the failing request is required to resolve this problem.
- **CORBA::IMP_LIMIT**
  This exception is raised if the data is too long to send over SNA, or there are too many active transactions in the server.
- **CORBA::INVALID_TRANSACTION**
  There is a problem with the transactional environment (transaction context) available to the request. Look at the minor code for more information.
- **CORBA::MARSHAL**
  This exception is raised when Component Broker receives unexpected status flags or data in an incorrect format from a tier-3 system.
- **CORBA::NO_IMPLEMENT**
  This exception is raised when a feature is not available in the current server.
- **CORBA::NO_PERMISSION**
  This exception is returned when a tier-3 system does not accept the userId/password sent with a PAA request.
- **CORBA::OBJECT_NOT_EXIST**
  This exception is returned when the transaction program name (TPN) value supplied on the APPC Connection is not an acceptable SNA value. The minor code raised with the exception indicates why the exception occurred.
- **CORBA::PERSIST_STORE**
  This is the exception passed to a Component Broker application when a PAA request fails. To discover the reason behind the failure, look in the activity log at the exceptions/messages that appear for the same thread just before the

CORBA::PERSIST_STORE occurs. Generally, the first error message/exception indicate the cause of the problem, and subsequent messages indicate the consequence of the failure.

- **CORBA::TRANSACTION_ROLLEDBACK**
  The transaction either has, or is about, to rollback. Therefore further operations on this transaction are not possible.
- **CORBA::TRANSIENT**
  This exception is raised when Component Broker is unable to exchange log names (XLN) with a tier-3 system. This is required before transactional requests can be sent to the tier-3 system.
- **CORBA::UNKNOWN**
  This exception is raised when a non-CORBA exception is caught by Component Broker.
- **CosTransactions::HeuristicCommit**
  This exception is raised if a tier-3 system commits a transaction without involving Component Broker's transaction service in the decision. If Component Broker has rolled the transaction back then heuristic damage has occurred.
- **CosTransactions::HeuristicHazard**
  This exception is raised if one or more tier-3 systems cannot be contacted when Component Broker it attempting to terminate (commit or rollback) a transaction. This means there is the possibility of heuristic damage.
- **CosTransactions::HeuristicMixed**
  This exception is raised when Component Broker detects heuristic damage during the termination of a transaction.
- **CosTransactions::HeuristicRollback**
  An APPC tier-3 system has rolled back the transaction independently from Component Broker. If Component Broker has committed the transaction, then heuristic damage has occurred.
- **CosTransactions::NoTransaction**
  This exception is raised if Component Broker is unable to continue with a request because a transaction is not active.
- **OTSAPPCConnection::AlreadyConnected**
  This exception is returned when an internal method is called out of turn. This is usually an internal logic error in Component Broker. A trace of the Transaction Service during the failing request is required to determine the cause of this problem.
- **OTSAPPCConnection::BadCommunications**
  This exception is raised when a tier-3 system sends incorrect data or status to Component Broker.
- **OTSAPPCConnection::CommunicationFailure**
  This exception is raised when an SNA communication failure is detected. This can be caused by an SNA network failure, shutdown of the tier-3 system or stopping the IBM Communication Server node locally.
- **OTSAPPCConnection::BadLocalLU**
  This exception is returned when the local LU name value supplied on the APPC Connection is not an acceptable SNA value. The minor code raised with the exception indicates why the exception occurred.
- **OTSAPPCConnection::BadModeName**
  This exception is returned when the mode name value supplied on the APPC Connection is not an acceptable SNA value. The minor code raised with the exception indicates why the exception occurred.

- **OTSAPPCConnection::BadPartnerLU**
  This exception is returned when the partner LU name value supplied on the APPC Connection is not an acceptable SNA value. The minor code raised with the exception indicates why the exception occurred.
- **OTSAPPCConnection::BadSecurity**
  This exception is returned when either the userId or password values supplied in the user's logon information is not an acceptable SNA value. The minor code raised with the exception indicates why the exception occurred.
- **OTSAPPCConnection::BadTPN**
  This exception is returned when the transaction program name (TPN) value supplied on the APPC Connection are not acceptable SNA values. The minor code raised with the exception indicates why the exception occurred.
- **OTSAPPCConnection::PartnerNotRecoverable**
  This exception is raised if a tier-3 system does not recognize the exchange log names (XLN) requests required before transactional requests can be sent to the tier-3 system. This means that either the tier-3 system does not support incoming transactional requests, or alternatively, a component in the tier-3 system used to receive transactional requests over SNA (for example an Encina PPC gateway) is either not running or incorrectly configured.
- **OTSAPPCConnection::NotConnected**
  This exception is returned when an internal method is called out of turn. This is usually an internal logic error in Component Broker. A trace of the Transaction Service during the failing request is required to determine the cause of this problem.
- **OTSAPPCConnection::SecurityNotValid**
  This exception is returned when a tier-3 system does not accept the userId/password sent with a PAA request.
- **OTSAPPCConnection::TPNUnavailable**
  This exception is raised if a tier-3 transaction program is not available.

**RELATED REFERENCES**

"Chapter 2. Activity Log for Problem Determination" on page 3

# Appendix F. SNA Data Formats

This file gives an overview of the format of data and other records that flow over SNA networks. The complete definition is documented in the IBM Manual *SNA Formats (GA27-3136)*.



## Transmission Headers (TH) and Request/Response Headers (RH)

The blocks of data that appear in an IBM Communication Server I-frames trace have the following structure:

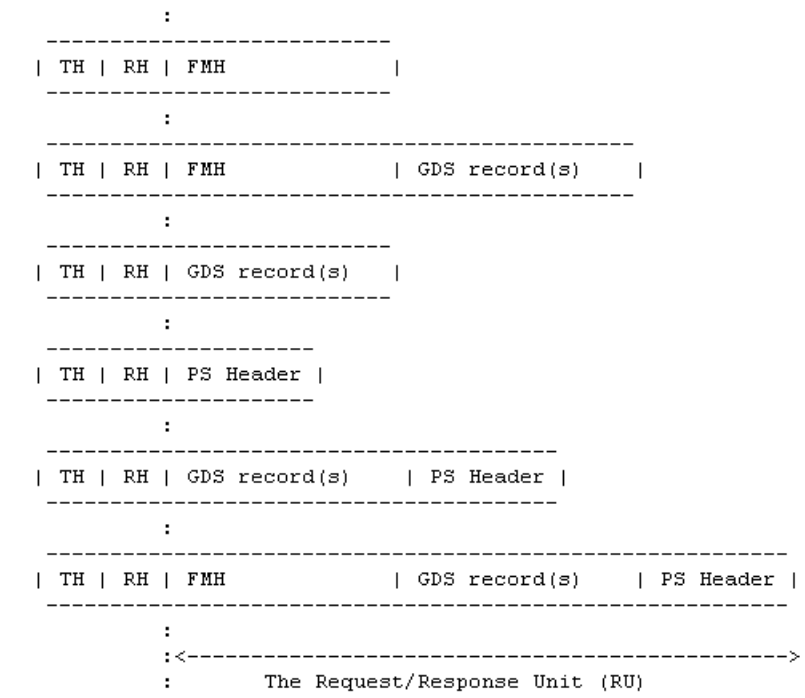
```
 ---------------------------
| TH | RH |    RU            |
 ---------------------------

 TH  Transmission Header
 RH  Request/Response Header
 RU  Request/Response Unit
```

The first six bytes are the Transmission Header (TH) which describes the source and destination of the message plus a sequence number. After the TH is the Request/Response Header or RH. This is three bytes of bit switches describing the type of data that follows and a number of indicators which reflect the type of request issued by the partner system/application. These indicators are used to set the state of the conversation.

| | |
|---|---|
| 0x800000 | 0=request, 1=response |
| 0x600000 | Two bits indicating:<br>00 An FMH is at the start of the RU<br>01 The RU contains Network control (NC) data<br>10 The RU contains Data Flow Control (DFC) data<br>11 The RU contains a request to start (bind) or stop (unbind) a session |
| 0x002000 | This is the Request Definite Response (RQD2) indicator and means a CONFIRMED response is requested. |
| 0x000020 | The Change Direction (CD) indicator means the sender is now waiting to receive data. |
| 0x000001 | The Conditional End Bracket (CEB) indicator usually means that this is the last flow for the conversation. |

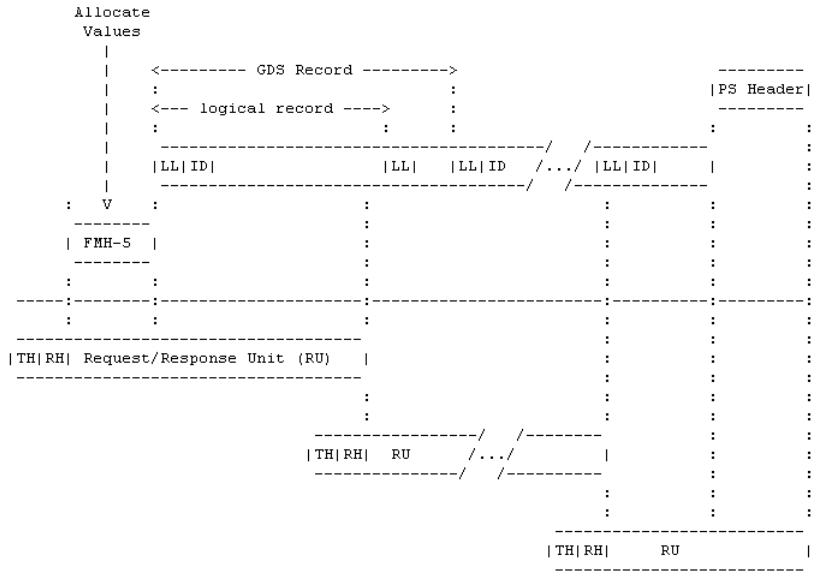
Finally, after the RH, is the Request/Response Unit (RU) which contains one or more types of SNA records. For example:




FMH is the Function Management Header.
PS Header is the Presentation Services Header.
GDS Records is the Generalized Data Stream Records.

As far as the Component Broker is concerned, if there is an FMH in the RU, then it appears first. After that, there may be one or more GDS records. If the conversation is synclevel 2, a single PS header may be attached at the end.

The responsibility for marshalling these records into RUs is split between Component Broker's OTS-APPC component and IBM Communications Server. The OTS-APPC builds GDS records and PS Headers. These are passed to IBM Communication Server in a buffer. IBM Communication Server builds FMHs and takes the data provided by OTS-APPC and breaks it down into RUs. The process of formatting RUs pays no interest to the boundaries of the records. The aim is to squeeze as much data into an RU as is allowed by the "max RU size" specified in the mode definition for the session.

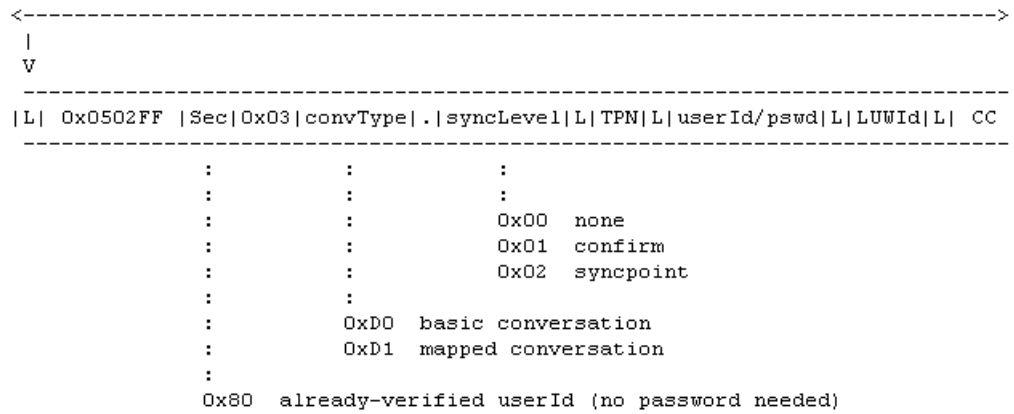



TH is the Transmission Header.
RH is the Request/Response Header.

**RELATED REFERENCES**
"APPC I-Frames Trace Samples" on page 178
"APPC Verb Trace Samples" on page 167
"Appendix F. SNA Data Formats" on page 207

## Attach FMH-5

The information to start a conversation is supplied by the application on the ALLOCATE verb. This is packed into an Attach Function Management Header (FMH) 5. The layout of an attach FMH-5 is shown below. See "APPC I-Frames Trace Samples" on page 178 for additional information.




CC = Conversation Correlator

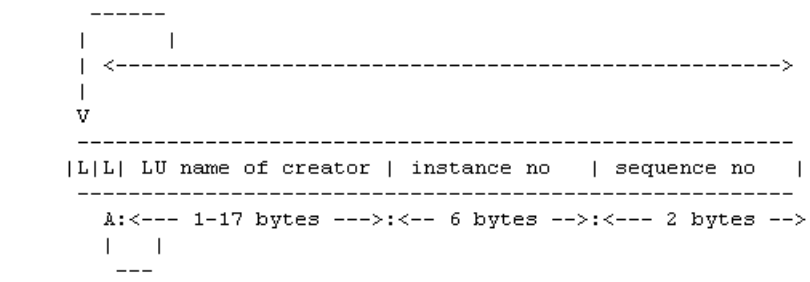
**RELATED REFERENCES**
"APPC I-Frames Trace Samples" on page 178

“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

## Logical Unit of Work Identifier (LUWId)

Logical Unit of Work Identifier (LUWId) is an identifier for a group of related pieces of work. The protected LUWId is used to group all recoverable work related to a single Component Broker transaction. So it can be thought of as the SNA equivalent of the CORBA CosTransactions::otid_t. In addition, there is an unprotected LUWId which is used to group together the non-recoverable work (that is, the first part of a pessimistic transaction) for a Component Broker transaction. Use of the unprotected LUWId is not mandatory but it can be used for accounting purposes. The two LUWIds associated with a Component Broker transaction are passed to IBM Communication Server in the SET_TP_PROPERTIES verb. They can also be seen on the GET_ATTRIBUTES verb.

The format of an LUWId is shown below:

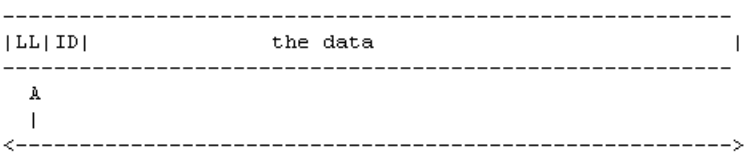



LUWIds flow across the SNA network in attach FMH-5 records and compare states records.

**RELATED REFERENCES**
“APPC I-Frames Trace Samples” on page 178
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

## GDS Records and Data Mapping

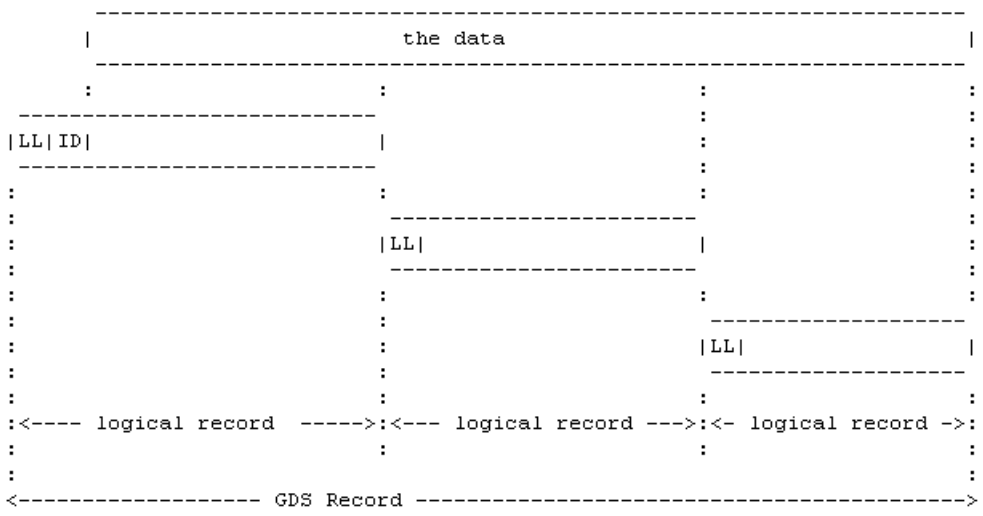
In order for the SNA network to keep track of the length of a piece of data, the data is sent around the network in GDS records (or GDS variables). In the simplest case these records have a two-byte **inclusive** length called the LL, a two-byte ID field that describes the type of data (see table below) and then the data itself.



The ID values can be:

| ID | Description | Comments |
|---|---|---|
| 0x12FF | Application data | Used by Component Broker to send data and CRUD requests to the tier-3 system. |
| 0x12F2 | User Controlled data | Use to send EXEC CICS LINK requests to a CICS tier-3 system. Not currently used by Component Broker for APPC support. |
| 0x1211 | Exchange Log Names (XLN) request/response | Used by Component Broker's transaction service to check the tier-3 system's transaction log is correct. |
| 0x1213 | Compare states request/response | Used by Component Broker's transaction service to resynchronize transactions after a server/network crash. |
| 0x1210 | Change Number of Sessions (CNOS) request/response | Used by two SNA LUs to negotiate the number of sessions that should be bound between them for a particular mode name. This process is controlled by IBM Communication Server on behalf of the Component Broker server using information from the mode definition. |

It is possible that the length of the data is more than can be represented by a two byte length. So a GDS record is actually made up of one or more concatenated logical records. The diagram below shows the data split over 3 logical records. Notice the ID field only appears in the first logical record. The top bit of each LL is a concatenation bit. When it is set, it means the following logical record contains data that is part of the same GDS record as this logical record. So in the diagram below, the concatenation bit is set in the first two logical records and not in the third.



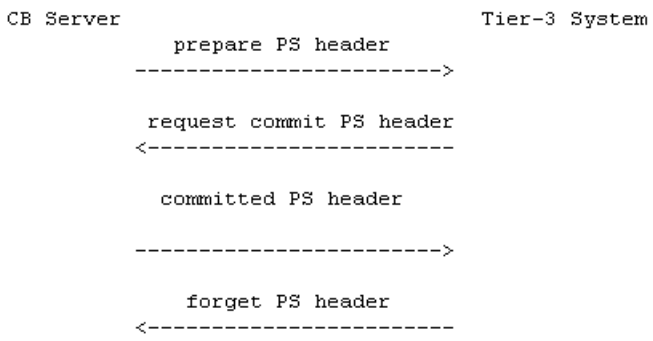
GDS records are passed between Component Broker and IBM Communication Server using the SEND_DATA and RECEIVE_AND_WAIT verbs.

**RELATED REFERENCES**

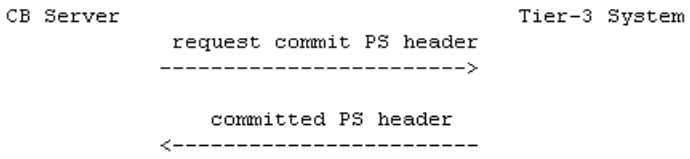
“APPC Verb Trace Samples” on page 167
“APPC I-Frames Trace Samples” on page 178
“Appendix F. SNA Data Formats” on page 207

## Presentation Services (PS) Header 10

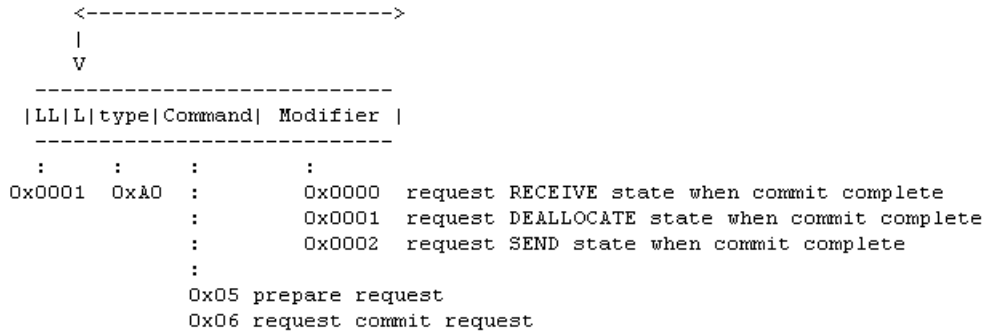
Presentation Services (PS) headers (type 10) are used to control the commit of a transaction across an SNA network. For example, if a Component Broker transaction has more than one synclevel 2 conversation, it uses a two phase commit on each conversation, as follows:




Alternatively, if only one synclevel 2 conversation is associated, a one phase commit is used.




The format of the PS header varies slightly. Component Broker sends prepare and request commit headers with a modifier of "DEALLOCATE" to indicate that the tier-3 system should end the conversation (by sending a Conditional End Bracket indicator) on the last PS header it sends. In addition, an IMS system will send a modifier of 0x0000 with its request commit PS header response to prepare. This has no meaning and is ignored.



Request commit PS headers sent by CICS, committed PS headers, and forget PS headers flow without a modifier. So does a heuristic mixed PS header which means the sender has committed part of the transaction and rolled back another.

```
 -----------------
|LL|L|type|Command|
 -----------------
  :     :     :
0x0001  0xA0  0x06  request commit request
              0x07  committed request
              0x08  forget request
              0x09  heuristic mixed request
```

Finally, there is a new LUWId PS header. It is included for completeness, but Component Broker does not send, nor does it expect to receive, this PS header.

```
 -------------------------------------
| LL | L | type | Command |  LUWId    |
 -------------------------------------
 :          :       :          :
 :          :       :          See LUWId
 :          :       :
 0x0001     0xA0    0x0A   new LUWId request
```

**RELATED REFERENCES**

“APPC I-Frames Trace Samples” on page 178
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

## Function Management Header 7 (FMH-7)

The Function Management Header (FMH) 7 is used for sending errors, backout requests, and abnormally terminating conversations. Its format is as follows:

```
<------------------->
 |
 V
 ------------------------------------------------------------------
|L|0x07|senseCode|ld|  Log data GDS record (optional)                |
 ------------------------------------------------------------------
           :      :
           :      0x00  No log data follows
           :      0x01  Log data follows
           :
           0x1008600B  A resource failed
           0x10086021  The TPN supplied on the allocate is not recognized
                       by the partner system
           0x10086031  PIP data not allowed (PIP data is not used by CB)
           0x10086032  PIP data is invalid
           0x10086034  Bad conversation type
           0x10086041  Bad Synclevel or XLN required (try restarting
                       IBM Communication Server or the CB server)
           0x080F60F1  UserId and/or Password not accepted by partner
           0x08240000  Backout (rollback) request
           0x08240001  Backout (rollback) request - resync pending locally
           0x084B6031  Tier-3 TPN not available
           0x084C0000  Tier-3 TPN not available (no retry)
           0x08640000  Conversation abended by program
           0x08640001  Conversation abended by partner system
           0x08640002  Conversation abended by system due to timeout
           0x08890000  Program issued SendError (not allowed with CB)
           0x08890001  Program issued SendError (not allowed with CB)
           0x08890100  Partner system issued SendError (not allowed with CB)
           0x08890101  Partner system issued SendError (not allowed with CB)
```

See “APPC I-Frames Trace Samples” on page 178 for examples of the trace.

Most FMH-7s are accompanied by a Conditional End Bracket (CEB) indicator. The exceptions are:

```
0x08240000  Backout (rollback) request
0x08240001  Backout (rollback) request - resync pending locally
```

which are accompanied by a Request Definite Response (RQD2) indicator and

```
0x08890000  Program issued SendError (not allowed with CB)
0x08890001  Program issued SendError (not allowed with CB)
0x08890100  Partner system issued SendError (not allowed with CB)
0x08890101  Partner system issued SendError (not allowed with CB)
```

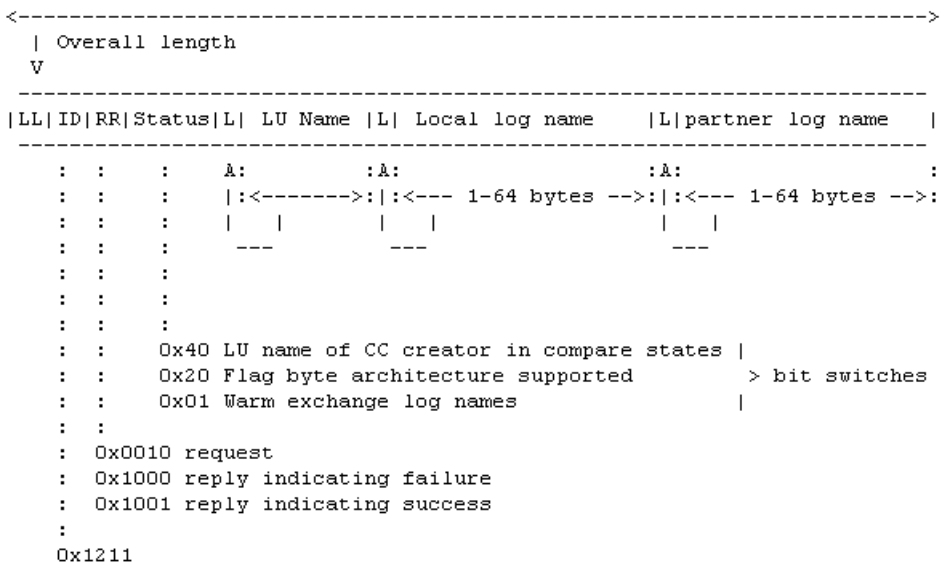
which do not automatically have any indicator with them. See “Transmission Headers (TH) and Request/Response Headers (RH)” on page 207 for more information on RQD2 and CEB.

**RELATED REFERENCES**

“APPC I-Frames Trace Samples” on page 178
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

## Exchange Log Names (XLN) GDS Record

The exchange log names (XLN) GDS record is used by an SNA LU (for example, a Component Broker server or a tier-3 system) to send information about the transaction log file it is using. XLN GDS records are sent on a conversation between the resync TPs (0x06F2) from each LU. See “APPC I-Frames Trace Samples” on page 178 for some examples of this trace.

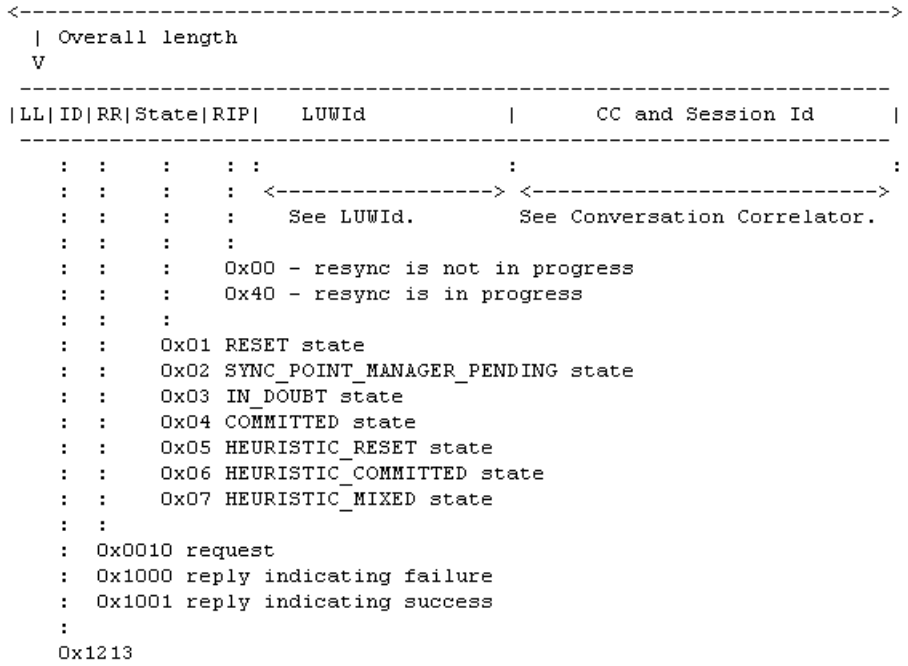



**RELATED REFERENCES**

"APPC I-Frames Trace Samples" on page 178
"APPC Verb Trace Samples" on page 167
"Appendix F. SNA Data Formats" on page 207

## Compare States GDS Record

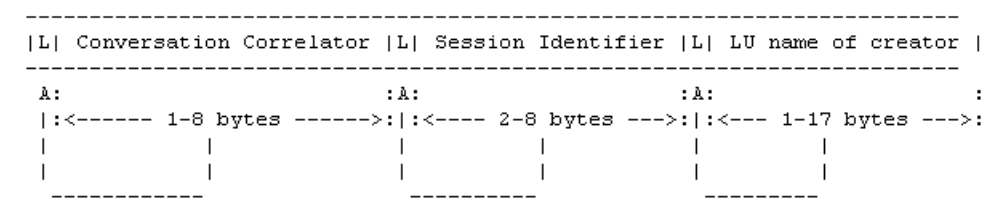
The Compare States GDS record is used by an SNA LU (for example, a Component Broker server or a tier-3 system) to send information about the state of a particular transaction. Compare States GDS records, optionally, follow XLN records on a conversation between the resync TPs (0x06F2) from each LU. See "Logical Unit of Work Identifier (LUWId)" on page 210 and "Conversation Correlator (CC) and Session Id" on page 216 for more information on their formats.



**RELATED REFERENCES**
“APPC I-Frames Trace Samples” on page 178
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

## Conversation Correlator (CC) and Session Id

A conversation correlator is an identifier for a particular conversation that is known by the LUs at either end of the conversation. It is assigned by the LU where the ALLOCATE (page 170) is issued, and it is sent in the attach FMH-5 preceded by a one byte exclusive length. It is also sent as part of a Compare States GDS record along with the Id of the session (requested by a GET_ATTRIBUTES call to IBM Communications Server) and optionally, the LU name of the LU where the conversation correlator was created (which is always the Component Broker).




**RELATED REFERENCES**
“APPC I-Frames Trace Samples” on page 178
“APPC Verb Trace Samples” on page 167
“Appendix F. SNA Data Formats” on page 207

# Appendix G. System Exceptions and Minor Codes

This section provides an explanation of the minor codes used by the Component Broker, the definition of the minor codes in hexadecimal order, and the code ranges for System Exceptions.

Each minor code begins with 0x4942, which is the OMG-assigned identification code for the Component Broker ORB. The remaining digits identify the minor code. In some cases, minor error codes may be reported without the vendor ID. Minor error codes reported from Java are in decimal and lack the vendor ID.

**Minor Codes**
Component Broker follows the CORBA model for exception handling. In this model, all exceptions can be associated with minor error codes. The minor error codes provide greater detail about the errors that can occur.

Error codes are used in several ways. They are returned in the minor code field of exception bodies (when appropriate), they are placed in the activity log as part of the PrimaryMessage, and they can be written in diagnostic messages on a computer screen. There is not a one-to-one mapping of exception names to minor error codes. Rather, a minor error code can be associated with several different exceptions. This means a minor error code message can mean different things depending on which exception was thrown.

The explanation of each minor error code includes:

| | |
|---|---|
| **Minor Code** | The hexadecimal value of the minor error code. |
| **Error Text** | The text string that identifies the minor error code. |
| **Explanation** | A description of the problem that caused the error. |
| **User Response** | Actions needed to resolve the problem, if appropriate. |

**Minor Code Messages**
The minor code messages in this section are organized in hexadecimal order, by minor code. Some multiple minor codes have the same hexadecimal code.

**0x49420031 SOMDERROR_OPSYS**
**Explanation:** An operating system error has occurred.
**User Response:** Check the log for more information.

**0x49420032 SOMDERROR_CouldNotLoadLibrary**
**Explanation:** Client initialization failed to load either security or work load management library or DLL.
**User Response:** Check the log for more information.

**0x49420033 SOMDERROR_NoMemory**
**Explanation:** A memory allocation failed.
**User Response:** Increase system resources. Check if process has a memory leak.

**0x49420034 SOMDERROR_NotImplemented**
**Explanation:** The invoked operation is not supported in the product or is not valid on the target object.

**User Response:** Check that the operation being invoked and the runtime type of the target object are compatible. Reference the documentation for the operation to find out about any restrictions.

**0x49420035 SOMDERROR_InvalidProtocolInformation**
**Explanation:** The configuration of the communications protocol(s) is incorrect. Supported communications protocols are TCP/IP and IPC.
**User Response:** Insure that at least one valid communications protocol image was configured using the Systems Management tool (for either TCP/IP or IPC). Insure that a host image was configured using the correct hostname. Insure that for each communications protocol configured, the csProfileTag and portNumber are set and that the portNumber is not in use by another process on the system. (The portNumber is the port on which the location-service daemon listens for requests.) The "profile tag" and "port number" settings must be unique for each communications protocol. Insure that for each server registered in the Implementation Repository, the set of supported communications protocols for that server intersects with the set of communications protocol images configured using the Systems Management tool.

**0x49420036 SOMDERROR_InvalidConfigSetting**
**Explanation:** A configuration setting or environment variable has not been properly set.
**User Response:** An error log entry indicates which configuration setting or environment variable is not properly set. If the reported variable is HOSTNAME, insure that a host image was configured using the Systems Management tool. If the reported variable is SOMCBASE, insure that the product was properly installed (SOMCBASE should be set during product installation to be the directory into which the product was installed.) If the reported variable is SOMCBENV, insure that SOMCBENV has one of the following forms: D:<image-name> S:<image-name> C:<image-name> A:<image-name> where "D:" is used when starting the location-service daemon, "S:" is used when starting a server process, "C:" is used when starting a client process, and "A:" is used when starting a systems-management agent process. The <image-name> is the name of a systems-management image. (For servers, the image name is the same as the server alias.) For non-managed clients, the SOMCBENV environment variable should instead be set to the name of a configuration file that contains configuration settings for the process. The default configuration file is installed in the "etc" subdirectory of the directory in which the product was installed, and is named "somcbenv.ini".

**0x49420036 SOMDERROR_SOMDDAlreadyRunning**
**Explanation:** The location-service daemon cannot begin listening because another process is already using the port number on which the daemon was configured to listen. The most likely cause of this error is attempting to start the location-service daemon process when another instance of the process is already running.
**User Response:** Do not attempt to start the location-service daemon, or terminate the other instance. If no other location-service daemon is running, try reconfiguring the location-service daemon to listen on a different port number. (Each communications protocol is configured with a separate port number using the Systems Management tool.)

**0x49420038 SOMDERROR_HostAddress**
**Explanation:** A failure to map a hostname (of a different machine) to a host address.

**User Response:** Ensure that the host with which this process is attempting to communicate is known and can be reached via TCP/IP. For example, try to `ping` the remote host by hostname.

**0x49420039 SOMDERROR_CouldNotStartProcess**
**Explanation:** The location daemon could not start a server process.
**User Response:** Check the log for more information.

**0x4942003A SOMDERROR_CouldNotStartThread**
**Explanation:** A thread could not be started.
**User Response:** Check the log for more information. Increase system resources.

**0x4942003B SOMDERROR_NoMessages**
**Explanation:** No request messages are pending in a server process when the server invoked CORBA::BOA::execute_next_request or CORBA::BOA::execute_request_loop with the CORBA::BOA::SOMD_NO_WAIT flag.
**User Response:** Wait for a request to become available, or use the CORBA::BOA::SOMD_WAIT flag when calling CORBA::BOA::execute_next_request or CORBA::BOA::execute_request_loop.

**0x4942003C SOMDERROR_MarshalingError**
**Explanation:** An error has occurred when trying to marshall or demarshall method parameters or return results as part of a remote invocation. This occurs, for example, if the process attempts to pass a IOM proxy as a method parameter or return result (only objects that inherit from CORBA::Object_ORBProxy can be passed on cross-process invocations). This error can also occur when demarshalling an inout sequence, if the length of the incoming sequence is greater than the original sequence maximum. This error can also occur when using the Dynamic Skeleton Interface (DSI), if methods are not invoked on the ServerRequest object in the correct order.
**User Response:** Ensure that IOM proxies are not passed as method parameters or return results. Ensure that inout sequences do not grow beyond the maximum of the sequence. If using the DSI, ensure that operations are invoked on the ServerRequest object in the correct order.

**0x4942003D SOMDERROR_CommTimeOut**
**Explanation:** A process has timed out waiting for a response from another process.
**User Response:** Ensure that the other process is still active. (Typically a client receives this error when the server has terminated or hung due to an application error.) To increase the timeout period, change the "request timeout" setting for the process using Systems Management. (The default setting is 30 seconds.) Setting "request timeout" to zero results in an infinite timeout.

**0x4942003E SOMDERROR_CannotConnect**
**Explanation:** A client process was unable to connect to a server process when attempting to invoke a method on a proxy to an object residing in that server process.
**User Response:** Ensure that the location-service daemon is running on the machine on which the server resides. Ensure that the object reference is still valid. Ensure that the two machines are connected (for instance, try to `ping` the remote machine).

**0x4942003F SOMDERROR_No_Server_Available**
**Explanation:** A client has invoked a method on a proxy to an object residing in a server group, but no server in the server group is currently available or the server

group cannot be reached.
Either a method call was made on a server group aware object, but the server group has no servers configured in it or a method call was made on a server group aware object for which there is at least one server configured, but none of the servers that are available were selected by the configured bind policies.
**User Response:** Configure at least one server into the server group if the server group has not servers configured in it. If the server group has at least one server configured, then investigate the possibility that the configured bind policies are deselecting all the available servers. This may be the correct behavior in this particular situation or the bind policy may need to be modified.
Alternatively, there may be a problem communicating with one or of the servers so check that the servers in the server group are running and that communication is possible between the client or server that raised this error and the servers in the server group and then shut down and restart the client or server that raised this error and/or re-initialize the application that hit the error. Alternatively, catch the error and retry the method call repeatedly until a server becomes available, but consider the possibility of a permanent server failure or communications failure.

**0x49420040 SOMDERROR_BadObjref**
**Explanation:** An invalid object reference was used. For example, this error is sent from a server to a client if the server receives a reference to an object that no longer exists or cannot be located in that server. This error can occur in a client process if an invalid string is passed to CORBA::ORB::string_to_object. This error occurs in a server if CORBA::BOA::create is called with input ReferenceData that doesn't map to any known exportable object residing in that server. The error also occurs if CORBA::BOA::get_id is invoked on a nil object reference or on an object reference that has no associated ReferenceData in that server. The error also occurs if a server attempts to export an object reference that has no associated ReferenceData in that server, or if a non-server attempts to pass a local object as a parameter on a remote method invocation. (A "non-server" is any process that has not yet called CORBA::BOA::impl_is_ready.)
**User Response:** In a client process, insure that the object to which the object reference refers still exists. Insure that strings passed to CORBA::ORB::string_to_object have not been corrupted or truncated. (There is no maximum length for an object reference string; some are larger than others.) Insure that servers don't attempt to export objects that aren't handled by the application adapter of the server. Also, IOM proxies cannot be exported from a server.

**0x49420041 SOMDERROR_Unknown**
**Explanation:** An unexpected error occurred during an operation.
**User Response:** Report the occurrence to technical support.

**0x49420042 SOMDERROR_CommunicationsError**
**Explanation:** A communications failure occurred. For example, a process could have received an unknown or unexpected message type or message content, the process could have encountered a low-level communications failure in attempting to send a message or binding to a socket, or an unexpected broken connection could have occurred.
**User Response:** Ensure that communications resources are functioning properly. For instance, when using TCP/IP, try to `ping` the remote host. Ensure that the process with which this process is trying to communicate has not failed due to an application error.

**0x49420043 SOMDERROR_ImplRepIO**
**Explanation:** The Implementation Repository database cannot be accessed.

**User Response:** Ensure that the Implementation Repository was correctly created and configured using the Systems Management tool. Each host machine must have its own Implementation Repository.

**0x49420044 SOMDERROR_EntryNotFound**
**Explanation:** An entry in the Implementation Repository was not found when attempting to delete, update, or find it.
**User Response:** Insure that the specified server alias or UUID matches a server that was previously registered in the Implementation Repository.

**0x49420045 SOMDERROR_ClassNotFound**
**Explanation:** Conversion of an IOR to an object failed because the class name was unknown or it's proxy factory cannot be created.
**User Response:** Verify the class implementation and its bindings exist.

**0x49420046 SOMDERROR_ServerAlreadyExists**
**Explanation:** A server was unable to register with the location-service daemon during CORBA::BOA::impl_is_ready. This is typically caused by another server already being registered with the location-service daemon under the given server UUID. (Only one instance of a particular server can be running on a given host at once.)
**User Response:** Terminate the duplicate server process. If no duplicate server process is running, restart the location-service daemon.

**0x49420047 SOMDERROR_CtxNoPropFound**
**Explanation:** A specified CORBA::Context property was not found. For example, if an invalid property name was passed to CORBA::Context::delete_values, this error occurs.
**User Response:** Ensure that the specified property name exists in the Context object.

**0x49420048 SOMDERROR_BadParm**
**Explanation:** An application supplied an invalid parameter to an operation.
**User Response:** The error log should contain a message indicating which operation was given the invalid parameter. Consult the documentation for that operation and insure that the parameters passed to it are valid.

**0x49420049 SOMDERROR_AuthnFail**
**Explanation:** An application attempted to manipulate an entry in the Implementation Repository for a server that is either being managed or disabled by the Systems Management tool. (Such entries in the Implementation Repository cannot be deleted or updated using the ImplRepository programmatic interface. Only entries registered using the ImplRepository interface can be updated or deleted using the ImplRepository interface.) This error is also raised if CORBA::ImplRepository::find_impldef is used to find a server that was disabled by Systems Management.
**User Response:** Use the Systems Management tool to manipulate the server. Insure that the server has not been disabled by the Systems Management tool. Insure that all entries in the Implementation Repository to be updated or deleted programmatically were originally added programmatically (rather than via the Systems Management tool).

**0x4942004A SOMDERROR_DuplicateEntry**
**Explanation:** The application attempted to add a duplicate entry to the Implementation Repository, or attempted to update the server alias of an existing entry using a name that is not unique.

**User Response:** Ensure that the server UUID and server alias of the ImplementationDef to be added or updated in the Implementation Repository are unique. (The server alias need not be unique throughout the network but must be unique in each Implementation Repository.)

**0x4942004B SOMDERROR_Internal**
**Explanation:** Unknown.
**User Response:** Report the occurrence to technical support.

**0x4942004D SOMDERROR_WrongRefType**
**Explanation:** An object reference of the wrong type was used. The most common reason for this error to occur is if a client invokes an operation on an object in a server and the object does not support the invoked method. (To support a given operation, a server must have been compiled and linked with the server-side C++ bindings for the interface that introduces that IDL operation.) Another situation in which this error can occur is if a server application invokes CORBA::BOA::get_id and passed in a proxy object (rather than a local object).
**User Response:** Ensure that a server is compiled and linked with all the server-side C++ bindings for the interfaces it exports. Ensure that a server does not pass a proxy object to CORBA::BOA::get_id.

**0x4942004E SOMDERROR_SOMDDNotRunning**
**Explanation:** A server was unable to register with the location-service daemon (in CORBA::BOA::impl_is_ready), because it was unable to contact the daemon. This is usually a result of the daemon not running or the daemon running on a different port number than the server expects.
**User Response:** Ensure that the location-service daemon is running on the same host as the server. Ensure that the port number configuration setting for each communications protocol is the same for the systems-management server image and daemon image.

**0x49420051 SOMDERROR_DataConversion**
**Explanation:** Failure to perform code-set translation of character data occurred. This could result from a failure of the XPG4 functions iconv() or nl_langinfo(). It can occur if the process is using a non-standard XPG4 code set that does not map to an OSF code set. It can occur if the native code set for the process (as reported by the XPG4 function nl_langinfo) does not match the nativeCharSet configuration data of the process (configured using the Systems Management tool). It can occur if a server does not have XPG4 code set converters for the transmission code set chosen by the client process. It can occur if the "char code sets" configuration setting for the server contains one or more code sets for which the process cannot open (using iconv_open) XPG4 converters. It can occur in a client process when attempting to communicate with a server if there is no common code set between the client and the server.
**User Response:** Ensure that, when using the translationEnabled configuration setting, that the other NLS-related configuration settings have been set correctly. Also ensure that the correct XPG4 code-set converters have been installed and that all environment variables (such as LOCPATH) required by XPG4 have been set properly. Ensure that both client and server are using standard code sets (recognized by both XPG4 and OSF) and that there is some code set supported by both the client and the server.

**0x49420052 SOMDERROR_IRIncoherent**
**Explanation:** An Interface Repository object references another named Interface Repository object which no longer resides in the IR database.
**User Response:** Call your IBM customer support center and report the problem.

**0x49420053 SOMDERROR_IRInternal**
**Explanation:** An internal programming or database error has occurred.
**User Response:** Call your IBM customer support center and report the problem.

**0x49420054 SOMDERROR_IRDuplicateEntry**
**Explanation:** An attempt was made to create an Interface Repository object and one already exists in the Interface Repository with either the same CORBA::RepositoryId or the same name within that container.
**User Response:** Change the "id" (CORBA::RepositoryId) parameter that is passed to the 'create_xxxx' operation, or change the "id" (CORBA::RepositoryId) value of the object already in the IR which is causing the duplicate entry error via the "id" write operation, or change the name of one of the two conflicting objects within that container.

**0x49420055 SOMDERROR_IREntryNotFound**
**Explanation:** A "create_xxxx" operation determined that one of the input parameters referenced an Interface Repository object which was not found in the database.
**User Response:** Specify a named object that does exist in the Interface Repository database.

**0x49420056 SOMDERROR_IRCannotConnect**
**Explanation:** The Interface Repository database could not be found or accessed properly. If this error is thrown, it occurs during a call to "resolve_initial_references" (with an input string of "InterfaceRepository").
**User Response:** Ensure that the Interface Repository database exists and is configured properly. Ensure that the directory/file permissions associated with the Interface Repository database allow access by the user receiving the exception.

**0x49420057 SOMDERROR_IRInUse**
**Explanation:** Another thread or process is already updating that portion of the Interface Repository database.
**User Response:** Retry the Interface Repository operation that generated the exception at a later time.

**0x494200C8 SOMSG::CorruptServiceContext**
**Explanation:** A server groups service context was corrupted.
**User Response:** Check for other symptoms of a transmission error. Gather information about the problem and follow your local procedures for reporting problems.

**0x494200C8 SOMSG::MCIncompleteConfig**
**Explanation:** The server groups code has failed to find some of its data in the Common Data Store.
**User Response:** Delete and reinstate the configuration using the Systems Management User Interface.

**0x494200C8 SOMSG::MCNCRootNotFound**
**Explanation:** The server groups code has failed to find the root naming context of the name tree.
**User Response:** Check for other symptoms of a naming service error, then gather information about the problem and report the occurrence to technical support.

**0x494200C8 SOMSG::MCStateInvalid**
**Explanation:** The server groups code has detected an invalid state in the Common

Data Store.
**User Response:** Delete and reinstate the configuration using the Systems Management User Interface.

**0x494200C8 SOMSG::NoServerDefined**
**Explanation:** A method call was made on a server group aware object, but the server group has no servers configured in it.
**User Response:** Configure at least one server into the server group.

**0x494200C8 SOMSG::ObjrefAddFailure**
**Explanation:** The server groups code has failed to add a server object reference into an internal collection, possibly because of a shortage of memory.
**User Response:** Increase the amount of memory available and retry.

**0x494200C8 SOMSG::RebindMinorCode**
**Explanation:** A server that is a member of a server group has detected that the client needs to refresh its copy of certain configuration information relating to the server group.
**User Response:** No action is required as this minor code is used in normal processing.

**0x494200C9 SOMSG::MCNSNotAvailable**
**Explanation:** The server groups code has detected an unexpected error when using the naming service.
**User Response:** Check for other symptoms of a naming service error, then gather information about the problem and report the occurrence to technical support.

**0x494200C9 SOMSG::MCServerNotMember**
**Explanation:** A server has attempted to register or deregister as a member of a server group, but the server is not configured as a member of the server group.
**User Response:** Check the configuration to ensure that the server is not a member of the server group. Try stopping the server and restarting it. If the error still occurs, gather information about the problem and follow your local procedures for reporting problems.

**0x494200C9 SOMSG::NoServerAvailable**
**Explanation:** A method call was made on a server group aware object for which there is at least one server configured, but none of the servers that are available were selected by the configured bind policies.
**User Response:** Investigate the possibility that the configured bind policies are deselecting all the available servers. This may be the correct behavior in this particular situation or the bind policy may need to be modified. Alternatively, there may be a problem communicating with one or of the servers so check that the servers in the server group are running and that communication is possible between the client or server that raised this error and the servers in the server group and then shut down and restart the client or server that raised this error and/or re-initialize the application that hit the error. Alternatively, catch the error and retry the method call repeatedly until a server becomes available, but consider the possibility of a permanent server failure or communications failure.

**0x494200CA SOMSG::MCCDSError**
**Explanation:** The server groups code has detected an unexpected error while accessing the Common Data Store.
**User Response:** Delete and reinstate the configuration using the Systems Management User Interface.

**0x494200CA SOMSG::UnknownServer**
**Explanation:** The object request broker has attempted to send a request on a server group aware object to a particular server but the request has failed and the server groups code no longer recognizes the server as a member of the server group in spite of the fact that it was the server groups code that selected the server from the members of the server group.
**User Response:** Gather information about the problem and follow your local procedures for reporting problems.

**0x494200CB SOMSG::MCInvalidUse**
**Explanation:** An unimplemented method in an internal server groups object was called.
**User Response:** Check that you are not making calls to internal server groups objects. If not, then gather information about the problem and follow your local procedures for reporting problems.

**0x494200F9 SOMGWDefs::GatewayNotImplemented**
**Explanation:** A non-server group enabled client has attempted to make a method call on a server group aware object. The method call was sent to the server group gateway server and was rejected since the gateway server is not yet implemented to forward requests to the server group.
**User Response:** A non-server group enabled client has attempted to make a method call on a server group aware object. The method call was sent to the server group gateway server and was rejected since the gateway server is not yet implemented to forward requests to the server group.

**0x494200FA ICACHE_BAD_INV_ORDER_KEYINIT**
**Explanation:** Key not set prior to cache operation.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x494200FA ICACHE_BAD_PARAM_INVATTR**
**Explanation:** An invalid attribute name was passed on a DAO (Data Access Object) call.
**User Response:** Make sure that the data object DLL and the system management DDL are consistent.

**0x494200FA ICACHE_INTERNAL_CACHNTRY**
**Explanation:** A set to get attribute value was called from a cache entry which is to be deleted.
**User Response:** Check the error log for more information.

**0x494200FA ICACHE_NO_IMPLEMENT_DATATYPE**
**Explanation:** A database attribute type not implemented by the cache component was encountered in processing a DAO (Data Access Object) call.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x494200FA ICACHE_UNKNOWN_DATATYPE**
**Explanation:** A database attribute type not implemented by the cache component was encountered in processing a DAO (Data Access Object) call.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x494200FA NO_MEMORY_DAO**
**Explanation:** The cache component is unable to acquire memory.

**User Response:** Increase the amount of virtual memory on the server system or reduce the number of other tasks running on the server system.

**0x494200FB ICACHE_BAD_INV_ORDER_CACHEMGRINIT**
**Explanation:** Application server internal error.
**User Response:** Check the installation of Component Broker and report problem to technical support.

**0x494200FB ICACHE_BAD_PARAM_INCOMPTYPE**
**Explanation:** The CORBA typecode used on a DAO call is inconsistent with the database attribute type.
**User Response:** Make sure that the data object DLL and the system management DDL are consistent with each other, and are also consistent with the table definition in the relational database.

**0x494200FB ICACHE_INTERNAL_CACHEMGR**
**Explanation:** A program error occurred in the cache component.
**User Response:** Report the occurrence to technical support.

**0x494200FC ICACHE_BAD_PARAM_KEYATTR**
**Explanation:** The key attribute name was passed on a setValue DAO call.
**User Response:** Make sure that the data object DLL and the system management DDL are consistent and correct.

**0x494200FC ICACHE_BAD_INV_ORDER_NODBCOMM**
**Explanation:** Cache internal error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x494200FC ICACHE_INTERNAL_AIXDAO**
**Explanation:** A program error occurred in the cache component.
**User Response:** Report the occurrence to technical support.

**0x494200FD ICACHE_BAD_INV_ORDER_NOMAP**
**Explanation:** Map metadata not set prior to cache operation.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x494200FD ICACHE_BAD_PARAM_INVMAP**
**Explanation:** An invalid map name was passed on a DAO call.
**User Response:** Make sure that the data object DLL and the system management DDL are consistent and correct.

**0x494200FD ICACHE_INTERNAL_HASH**
**Explanation:** A program error occurred in the cache component.
**User Response:** Report the occurrence to technical support.

**0x494200FE ICACHE_BAD_PARAM_NONKEYATTR**
**Explanation:** A non key attribute was passed on a set key operation.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x494200FE ICACHE_INTERNAL_CACHE**
**Explanation:** A program error occurred in the cache component.
**User Response:** Report the occurrence to technical support.

**0x494200FF ICACHE_BAD_PARAM_SMALLBUF**
**Explanation:** An insufficient buffer size was provided in a getValue DAO call to retrieve a database attribute of the DATE/TIME/TIMESTAMP type.
**User Response:** Make sure that the data object DLL and the system management DDL are consistent and correct.

**0x494200FF ICACHE_INTERNAL_GLOBAL_CACHE**
**Explanation:** A program error occurred in the cache component.
**User Response:** Report the occurrence to technical support.

**0x494200100 ICACHE_BAD_PARAM_UNEQUALBUF**
**Explanation:** An incorrect size string was provided in a setValue DAO call to modify a database attribute of the DATE/TIME/TIMESTAMP type.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x494200100 ICACHE_INTERNAL_SQLACCESS**
**Explanation:** A program error occurred in the cache component.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420101 ICACHE_BAD_PARAM_NOTEXTRACT**
**Explanation:** A string value from ::CORBA::Any that was passed in a getValue DAO call failed to be extracted.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x49420101 ICACHE_INTERNAL_SQLERROR**
**Explanation:** A system error occurred in the cache component when:

- The program ran out of free cursors (maximum 64); the application tried to access more than 64 tables within a transaction
- An unexpected error occurred in accessing tables.

**User Response:** Check the error log for more information.

**0x49420102 ICACHE_BAD_PARAM_INVDATATYPE**
**Explanation:** Make sure that the data object implementation is correct and consistent with the system management DDL.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x49420102 ICACHE_INTERNAL_SQLERROR_CONNECT**
**Explanation:** The cache component failed to establish a database connection.
**User Response:** Check the error log for more information.

**0x49420103 ICACHE_BAD_PARAM_LARGEANYSTR**
**Explanation:** The string buffer passed in the ::CORBA::Any on a setValue DAO call was larger than the database attribute to be modified.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x49420103 ICACHE_INTERNAL_NOPREFEC**
**Explanation:** A program error occurred in the cache component.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420104 ICACHE_BAD_PARAM_WRONGNUMKEYATTRS**
**Explanation:** The wrong number of key attributes were passed on a DAO setKeyAttributes or reactivateFromData call.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x49420104 ICACHE_INTERNAL_FKREL**
**Explanation:** A program error occurred in the cache component.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420105 ICACHE_BAD_PARAM_LARGEANYBINARY**
**Explanation:** The binary buffer passed in the ::CORBA::Any on a setValue DAO call is larger than the database attribute to be modified.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl.

**0x49420105 ICACHE_INTERNAL_BUILDDAOS**
**Explanation:** Internal error. Cache unable to allocate a DAO object.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420106 ICACHE_BAD_PARAM_INVINTERFACE**
**Explanation:** An invalid map name was passed on a DAO call.
**User Response:** Make sure that the data object implementation is correct and consistent with the system management ddl. Verify that the metadata is consistent with the database.

**0x49420106 ICACHE_INTERNAL_PREFETCH**
**Explanation:** A program error occurred in the cache component.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420107 ICACHE_BAD_PARAM_INVFKRELID**
**Explanation:** Invalid relationship identifier.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x49420108 ICACHE_BAD_PARAM_INVPARENTFKRELID**
**Explanation:** Invalid relationship identifier.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x49420109 ICACHE_BAD_PARAM_INVCHILDFKRELID**
**Explanation:** Invalid relationship identifier.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x4942010A ICACHE_BAD_PARAM_INVCHILDKEY**
**Explanation:** Invalid key on cache operation.
**User Response:** Check the DO implementation code and report problem to technical support.

**0x4942010A ICACHE_BAD_PARAM_INVOVERFLOW**
**Explanation:** An invalid data value was passed to cache service from Data Object.

**User Response:** If the object attribute maps to a decimal datatype in the database, either the value passed was invalid, or it overflowed the field in the database record.

**0x4942010B ICACHE_BAD_PARAM_INVCURSOR**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x4942010C ICACHE_BAD_PARAM_INVCURSOROP**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x4942010D ICACHE_BAD_PARAM_EMPTYCOLL**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x4942010E ICACHE_BAD_PARAM_BEGINCOLL**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x4942010F ICACHE_BAD_PARAM_ENDCOLL**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x49420120 ICACHE_BAD_PARAM_BETWEENELEM**
**Explanation:** Internal cache error.
**User Response:** Check the error log for more information and report the occurrence to technical support.

**0x4942012C SOMERROR_NULL_CHARACTER_IN_STRING**
**Explanation:** A CORBA string or wstring value contained a NULL character. Used as a minor code for the DATA_CONVERSION exception.
**User Response:** Ensure that CORBA string and wstring values do not contain NULL characters

**0x4942012D SOMERROR_CHARACTER_OUT_OF_RANGE**
**Explanation:** A CORBA string contained a non-ASCII character (a character whose ordinal value was greater than 255). Used as a minor code for the DATA_CONVERSION exception.
**User Response:**Ensure that CORBA string values do not contain non-ASCII characters. This problem usually arises in Java, since Java strings can contain non-ASCII characters.

**0x4942012E SOMERROR_STRING_TOO_LONG**
**Explanation:** A string containing too many characters was passed for a CORBA string or wstring argument that was declared with a bounded length. Used as a minor code for the MARSHAL exception.
**User Response:** Ensure that the length of strings passed does not exceed the maximum length for the string declared in IDL.

**0x4942012F SOMERROR_STRING_IS_NULL**
**Explanation:** A null was passed where a CORBA string or wstring value was expected. Used as a minor code for the DATA_CONVERSION exception.
**User Response:** Ensure that your program does not pass null for CORBA string or wstring arguments. An empty string must be passed instead.

**0x49420130 SOMERROR_CDS_ERROR**
**Explanation:** Not currently used.
**User Response:** None.

**0x49420131 SOMERROR_IMPL_NOT_IN_DLL**
**Explanation:** Not currently used.
**User Response:** None.

**0x49420132 SOMERROR_DLL_NOT_LOADED**
**Explanation:** Not currently used.
**User Response:** None.

**0x49420133 SOMERROR_JAVA_CLASS_NOT_LOADED**
**Explanation:** A Java class could not be located or loaded. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Consult the activity log for a message containing the name of the Java class that couldn't be loaded and ensure that it exists in CLASSPATH and can be loaded.

**0x49420134 SOMERROR_INTERFACE_NOT_FOUND**
**Explanation:** A CORBA interface could not be located by IOM. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Consult the activity log the a message containing the name of the interface that couldn't be found. Ensure that an implementation exists for the requested interface.

**0x49420135 SOMERROR_MESSAGE_NOT_UNDERSTOOD**
**Explanation:** A CORBA operation could not be located by IOM. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Try rebuilding your object implementation (C++ DLL or Java class). If the problem persists contact Component Broker support.

**0x49420136 SOMERROR_CLASS_NOT_LOCALONLY**
**Explanation:** An attempt was made to call _create on a C++ implementation that had not been compiled as "localonly". Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Ensure that the IDL file has a "#pragma meta X localonly" statement for the interface or that idlc was run with the "-mlocalonly" option.

**0x49420137 SOMERROR_JAVAVM_DLL_NOT_FOUND**
**Explanation:** IOM was unable to load the DLL containing the Java Virtual Machine. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Consult the activity log for a message containing the name of the DLL containing the Java Virtual Machine (tagged with the text "loadAndInitVM"). Ensure that the VM DLL name is correct (should be "javai.dll"), that a directory containing that DLL ($JAVA_HOME\bin for NT, $JAVA_HOME/lib/aix/native_threads for AIX) is in PATH (for NT) or LIBPATH (for AIX), and that the DLL can be loaded. Also ensure that the JDK is at the level that Component Broker requires.

**0x49420138 SOMERROR_JAVAVM_NOT_JNI_ENABLED**
**Explanation:** The Java Virtual Machine that IOM loaded does not support the Java Native Interface. Used as a minor code with OBJECT_NOT_EXIST.
**User Response:** Ensure that the you have installed the Java JDK at the level required by Component Broker and that the $JAVA_HOME\bin (for NT) or $JAVA_HOME/lib/aix/native_threads (for AIX) directory appears in your PATH (for NT) or LIBPATH (for AIX) before any other JDK.

**0x49420139 SOMERROR_JAVA_REFLECTION_FAILED**
**Explanation:** IOM was unable to use the Java Reflection API to obtain the list of methods implemented by a Java class. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Ensure that your JDK is correctly installed as described above and that your Java implementation .class files will load without error. Contact Component Broker support if the problem persists.

**0x4942013A SOMERROR_JAVA_HELPER_NOT_LOADED**
**Explanation:** IOM was unable to load the Helper class for a Java implementation. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Consult the activity log for a message containing the name of the helper class that couldn't be found. Ensure that the Helper .class exists and is reachable from CLASSPATH.

**0x4942013B SOMERROR_JAVAVM_NOT_STARTED**
**Explanation:** The JNI_CreateJavaVM call on the installed JDK did not complete successfully. This means that IOM was unable to initialize the Java Virtual Machine. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Ensure that the you have installed the Java JDK at the level required by Component Broker and that it initializes correctly (try running "java java.lang.String". You should get an error for "main is not defined". Any other result is incorrect).

**0x4942013C SOMERROR_THREAD_ATTACH**
**Explanation:** The JNI_AttachCurrentThread call on the installed JDK did not complete successfully. This means that IOM was unable to bind to the Java Virtual Machine. Used as a minor code for OBJECT_NOT_EXIST.
**User Response:** Ensure that the you have installed the Java JDK at the level required by Component Broker and that it initializes correctly (try running "java java.lang.String". You should get an error for "main is not defined". Any other result is incorrect).

**0x494201C3 EXERR_FLOATINGPOINTOVERFLOW**
**Explanation:** The conversion from IEEE Double to MVS Double resulted in an exponent outside the range of valid exponents.
**User Response:** Restrict usage of floating point values to those supported by both IEEE Double and MVS Double.

**0x494201C3 EXERR_INVALIDSTREAMIOBUFFERRESETPARM**
**Explanation:** The StreamIO buffer reset param was invalid.
**User Response:** Internal error, contact IBM support.

**0x494201C3 EXERR_INVALIDSTREAMPOLICY**
**Explanation:** The StreamIO buffer header contained an unrecognized value for the stream policy.
**User Response:** Internal error, contact IBM support.

**0x494201C3 EXERR_UNSUPPORTEDCODEPAGE**
**Explanation:** Either the source or destination code page, or both, as given in the StreamIO header, is NULL. This minor code could also indicate that the compiler's support for codeset id conversion has failed.
**User Response:** Ensure that the code page(s) are valid, non-NULL, and that your compiler supports the codeset id conversion.

**0x494201C4 EXERR_FLOATINGPOINTNOTCONFIGURED**
**Explanation:** The conversion between MVS/IEEE Floats or Doubles was unable to occur because the sourceFloatingPointFormatID and the destinationFloatingPointFormatID did not match one of the supported formats.
**User Response:** Internal error, contact IBM support.

**0x494201C4 EXERR_INVALIDSTREAMFORMAT**
**Explanation:** The stioFormat of the IStreamLocalObjectIBMImpl_StreamIO_Impl header was expected to be (Octet) IExtendedStream::IBM, but was not.
**User Response:** Internal error, contact IBM support.

**0x494201C4 EXERR_OBJECTTABLELIMITEXCEEDED**
**Explanation:** The number of objects in the object table has exceeded the upper limit.
**User Response:** Internal error, contact IBM support.

**0x494201C5 EXERR_CODEPAGENOTCONFIGURED**
**Explanation:** Either the source or destination code page, or both, as given in the StreamIO header, is an Undefined code page.
**User Response:** Ensure that the code page(s) are defined and available on the machine.

**0x494201C5 EXERR_EMPTYTABLESEQUENCE**
**Explanation:** There are no entries in the object table.
**User Response:** Internal error, contact IBM support.

**0x494201C5 EXERR_INVALIDENDIANFORMAT**
**Explanation:** The StreamIO buffer header contained an invalid endianFormat value.
**User Response:** Acceptable values are:

- (Octet)LittleEndian
- (Octet)BigEndian

This is most likely caused by an internal error. Contact IBM support.

**0x494201C6 EXERR_ENDIANNOTCONFIGURED**
**Explanation:** The Endian format ID's of the StreamIO's were invalid.
**User Response:** Ensure that both source and destinationEndianFormatID's are either BigEndian or LittleEndian. This is most likely caused by an IBM internal error. Contact IBM support.

**0x494201C6 EXERR_INVALIDFLOATINGPOINTFORMAT**
**Explanation:** The Stream IO header indicated an invalid floating point format.
**User Response:** Acceptable values are IEEE754 and MVS. This is most likely caused by an IBM internal error. Contact IBM support.

**0x494201C6 EXERR_OBJECTTABLESTATEERROR**
**Explanation:** Table did not contain a sufficient number of entries.
**User Response:** Internal error, contact IBM support.

**0x494201C7 EXERR_FLOATINGPOINTUNDERFLOW**
**Explanation:** The conversion from IEEE Double to MVS Double resulted in an exponent outside the range of valid exponents.
**User Response:** Restrict usage of floating point values to those supported by both IEEE Double and MVS Double.

**0x494201C7 EXERR_OBJECTTABLEIDSEQUENCEERROR**
**Explanation:** Ensure TableID value sequence is maintained.
**User Response:** Internal error, contact IBM support.

**0x494201C7 EXERR_STRINGNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201C8 EXERR_CANTINSTALLINTOOBJECTTABLE**
**Explanation:** Object table was expected to be NULL, but was not.
**User Response:** Internal error, contact IBM support.

**0x494201C8 EXERR_CHARNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201C8 EXERR_ICONVCALLFAILED**
**Explanation:** Code page conversion of aString failed.
**User Response:** Verify string contains valid characters for the target code page.

**0x494201C9 EXERR_CANTAPPENDEMPTYOBJECTTABLE**
**Explanation:** An empty object table cannot be appended to the current object table.
**User Response:** Internal error, contact IBM support.

**0x494201C9 EXERR_OCTETNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CA EXERR_UNSIGNEDLONGNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CB EXERR_UNSIGNEDSHORTNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CC EXERR_LONGNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CD EXERR_SHORTNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CE EXERR_FLOATNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201CF EXERR_DOUBLENOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201D0 EXERR_BOOLEANNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201D1 EXERR_OBJECTNOTFOUND**
**Explanation:** The tag read from the stream did not match the expected data type.
**User Response:** Check that the order of reads/writes in internalize/externalize is correct.

**0x494201D2 EXERR_INVALIDBOOLEANVALUEFOUND**
**Explanation:** The value of a boolean was neither recognized as ((CORBA::Boolean) 0), nor ((CORBA::Boolean) 1).
**User Response:** Check the object being read/written.

**0x494201D9 EXERR_READPASSEDENDOFSTREAM**
**Explanation:** The number of bytes requested to be read from the stioBufferSeq exceeds the number of bytes left in the stioBufferSeq.
**User Response:** Check that the order and number of reads/writes in internalize/externalize is correct.

**0x494201F5 LCERR_UNKNOWN_SEE_EVENT_LOG**
**Explanation:** LifeCycle made an invocation which resulted in an exception being thrown by another component.
**User Response:** Examine activity log entries prior to this to see if another exception was logged. Exception is CORBA::UNKNOWN.

**0x494201F6 LCERR_SCOPE_MAN_ALREADY_INITIALIZED**
**Explanation:** An undocumented internal method was invoked. Results will be unpredictable.
**User Response:** Do not invoke any non-documented methods. Exception is ILifeCycleLocalObjectImpl::AlreadyInitialized.

**0x494201F7 LCERR_SCOPE_STRING_IS_NULL**
**Explanation:** The Scope parameter, or the Scope constructed from the ScopeString parameter,
contains a NULL element.
**User Response:** Reconstruct the scope parameter so that elements are either specified or
left to the defaults. Note that in a ScopeString, any element not explicitly specified in the string is given a default value as follows:

- cell - *LOCAL
- workgroup - *LOCAL
- host - *LOCAL
- server - *ANY

- container - *ANY
- home - *ANY

It is the user's responsibility to ensure that the combination of specified elements and supplied defaults is valid. Exception is CORBA::BAD_PARAM.

**0x494201F8 LCERR_FAC_FIND_ALREADY_INITIALIZED**
**Explanation:** An undocumented internal method was invoked. Results will be unpredictable.
**User Response:** Do not invoke any non-documented methods. Exception is ILifeCycleLocalObjectImpl::AlreadyInitialized.

**0x494201F9 LCERR_FAC_FIND_LOCATION_IS_NULL**
**Explanation:** An internal method was invoked illegally and with a null parameter. Results will be unpredictable.
**User Response:** Do not invoke any non-documented methods. Exception is CORBA::BAD_PARAM.

**0x494201FA LCERR_SCOPE_ELEMENT_NOT_RECOGNIZED**
**Explanation:** The ScopeString parameter contains an element other than cell, workgroup, host, server, container, or home.
**User Response:** Remove the invalid element from the ScopeString. Exception is IExtendeLifeCycle::UnrecognizedScopeElement.

**0x494201FB LCERR_FAC_FIND_INVALID_KEY**
**Explanation:** The key used to find the factory is invalid.
**User Response:** Check that the key used in the find factory operation is correct. Exception is CosLifeCycle::NoFactory.

**0x494201FC LCERR_FACTORY_NOT_FOUND**
**Explanation:** The factory is not registered.
**User Response:** Check that the key used in the find factory operation is correct. Try using a factory_finder with a less restrictive scope. Verify that the registration of the expected factories did not result in any errors. Exception is CosLifeCycle::NoFactory.

**0x494201FD LCERR_SINGLE_LOC_ALREADY_INITIALIZED**
**Explanation:** An undocumented internal method was invoked. Results will be unpredictable.
**User Response:** Do not invoke any non-documented methods. Exception is ILifeCycleLocalObjectImpl::AlreadyInitialized.

**0x494201FE LCERR_MO_REGISTRATION_REQUEST_FAILED**
**Explanation:** The FactoryRepository failed to complete all or some part of the requested operation. Probable cause is a failure in the naming component.
**User Response:** Check the event log for additional information. Exception is ILifeCycleRepositoryAdmin::RequestFailed.

**0x494201FF LCERR_MO_UNREGISTRATION_REQUEST_FAILED**
**Explanation:** The FactoryRepository failed to complete all or some part of the requested operation. Probable cause is a failure in the naming component.
**User Response:** Check the event log for additional information. Exception is ILifeCycleRepositoryAdmin::RequestFailed.

**0x49420200 LCERR_SERVER_BRANCH_UNRETRIEVABLE**
**Explanation:** The FactoryRepository failed to complete all or some part of the

requested operation. Probable cause is a failure in the naming component.
**User Response:** Check the event log for additional information. Exception is ILifeCycleRepositoryAdmin::RequestFailed.

**0x49420201 LCERR_UNABLE_TO_CLEAR_ELEMENT**
**Explanation:** Most likely problem with naming code when attempting to clear the repository.
**User Response:** Check the event log for additional information. Exception is ILifeCycleRepositoryAdmin::RequestFailed.

**0x49420202 LCERR_A_SCOPE_ELEMENT_IS_NULL**
**Explanation:** An internal method was invoked illegally and with part of the scope structure as null. Results will be unpredictable.
**User Response:** Do not invoke any non-documented methods. Exception is CORBA::BAD_PARAM.

**0x49420203 LCERR_INVALID_SCOPE**
**Explanation:** The Scope parameter, or the Scope constructed from the ScopeString parameter, did not obey the rules for valid scopes. The rules are:

- For the Cell, Workgroup and Host elements of scope, once *LOCAL has been specified at a lower level, higher levels must also be *LOCAL.
- For the Server, Container and Home elements of scope, once *ANY has been specified a t a higher level, lower levels must also be *ANY.

**User Response:** Reconstruct the scope parameter so that it meets the specifications above. Note that in a ScopeString, any element not explicitly specified in the string is given a default value as follows:

- cell - *LOCAL
- workgroup - *LOCAL
- host - *LOCAL
- server - *ANY
- container - *ANY
- home - *ANY

It is the users responsibility to ensure that the combination of specified elements and supplied defaults is valid. Exception is IExtendedLifeCycle::InvalidScope.

**0x49420204 LCERR_OUT_OF_MEMORY**
**Explanation:** Machine has run out of virtual memory.
**User Response:** Cancel some of the applications currently running on the system. Exception is CORBA::NO_MEMORY.

**0x49420205 LCERR_MIXIN_PTR_IS_NULL**
**Explanation:** Internal problem with a Life Cycle object.
**User Response:** Contact IBM Representative. Exception is CORBA::BAD_PARAM.

**0x49420206 LCERR_INVALID_OBJECT**
**Explanation:** Object is in an invalid state of absent.
**User Response:** Attempt to restart the server. Contact IBM Representative. Exception is CORBA::INV_OBJREF.

**0x49420207 LCERR_INVALID_LOCATION**
**Explanation:** An invalid location object was passed in. The Location object must

inherit fromIManageble.
**User Response:** Ensure that the location object is properly created. Exception is CORBA::BAD_PARAM.

**0x49420208 LCERR_SYSTEMS_MANAGED**
**Explanation:** An attempt was made to remove a systems managed lifecycle object. These objects cannot be removed programmatically.
**User Response:** Remove the lifecycle object using the Component Broker Graphical Interface. Exception is CosLifeCycle::NotRemovable.

**0x49420209 LCERR_BAD_KEY_STRING**
**Explanation:** An invalid key string was a member of the criteria passed to the create_object method.
**User Response:** Correct the key string and retry the create_object invocation. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020A LCERR_BAD_RELATIVENAME**
**Explanation:** An invalid relative name was a member of the criteria passed to the create_object method.
**User Response:** Correct the relative name and retry the create_object invocation. The relative name must be a character string. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020B LCERR_BAD_CELL_VISIBLE**
**Explanation:** An invalid value was supplied for the "cell visible" member of the criteria passed to the create_object method.
**User Response:** Correct the "cell visible" value and retry the create_object invocation. Cell visible must be a CORBA::Boolean value. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020C LCERR_BAD_HOST_VISIBLE**
**Explanation:** An invalid value was supplied for the "host visible" member of the criteria passed to the create_object method.
**User Response:** Correct the "host visible" value and retry the create_object invocation. Host visible must be a CORBA::Boolean value. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020D LCERR_BAD_WORKGROUP_VISIBLE**
**Explanation:** An invalid value was supplied for the "workgroup visible" member of the criteria passed to the create_object method.
**User Response:** Correct the "workgroup visible" value and retry the create_object invocation. Workgroup visible must be a CORBA::Boolean value. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020E LCERR_BAD_SCOPE**
**Explanation:** An invalid scope was a member of the criteria passed to the create_object method.
**User Response:** Correct the scope value and retry the create_object invocation. The scope must be of type IExtendedLifeCycle::Scope. Exception is CosLifeCycle::InvalidCriteria.

**0x4942020F LCERR_BAD_SCOPE_STRING**
**Explanation:** An invalid scope string was a member of the criteria passed to the create_object method.

**User Response:** Correct the scope string value and retry the create_object invocation. The scope string must be a character string. Exception is CosLifeCycle::InvalidCriteria

**0x49420227 CORBAMinorNamingAlreadyBound**
**Explanation:** This indicate that an object is already bound to the name.
**User Response:** Re-binding operations unbind the name, then rebinds the name without raising this exception.

**0x49420227 CORBAMinorNamingCannotProceed**
**Explanation:** This indicate that the implementation has given up for some reason.
**User Response:** The client may be able to continue the operation using the returned naming context.

**0x49420227 CORBAMinorNamingContextObjectCreationFailed**
**Explanation:** The creation of a NamingContext object or the narrowing to a NamingContext implementation object failed.
**User Response:** Most likely system is running out of memory, or the remote server of this object resided is not responding.

**0x49420227 CORBAMinorNamingDCECDSError**
**Explanation:** This indicate a DCE error has occurred.
**User Response:** User can look at fsLocStat and CDS entry information in activity log to determine the problem.

1. If fsLocStat is of type CDS_UNKNOWNATTRIBUTE or CDS_INVALIDNAME, then the cds_attributes file does not contain the Naming attributes.
2. If fsLocStat is of type CDS_ACCESSDENIED, then DCE credential have been expired. Do dce_login to refresh your DCE server credentials.
3. If fsLocStat is of type CDS_UNKNOWNENTRY, then no object bound to the supplied name is found.
4. If fsLocStat is of type CDS_ERROR, 0x0DCE60D6, 0x0DCE6042 or CDS_CLEARINGHOUSEDOWN, then make sure all the DCE services are still running.

**0x49420227 CORBAMinorNamingHeapOutOfMemory**
**Explanation:** Attempt to allocate memory on the heap has failed.
**User Response:** System is running out of memory.

**0x49420227 CORBAMinorNamingInvalidCORBAName**
**Explanation:** This indicate that the name is invalid. (This exception may be raised upon further implementation restrictions.)
**User Response:** A name (of type CosNaming::Name) is valid if none of the following is true:

1. It's length field is not equal to 0.
2. It does not have a component where the id of the component is a NULL pointer or the value of this id is equal to a null-string.
3. It does not have a component where the kind of the component is a NULL pointer.

Additional to above, for CDS-based implementation, a name n is invalid if:

1. It does not follow the constraints on CDS global names.
2. It does not confirm or translate to the ISOLatin1 character set.

3. The total name length is greater than 400.
4. The length of a simple name is greater then 254.
5. A simple name starts with the character '/'.
6. The simple name following "/.:" contains with the character '='.

**0x49420227 CORBAMinorNamingListSizeExceedMaxValue**
**Explanation:** The list size exceeds the allowed maximum value of 1000.
**User Response:** Make sure do not exceed the maximum value.

**0x49420227 CORBAMinorNamingNoImplementation**
**Explanation:** The invoked operation is not supported in the product or is not valid on the target object.
**User Response:** Check that the operation being invoked and the runtime type of the target object are compatible. Reference the documentation for the operation to find out about any restrictions.

**0x49420227 CORBAMinorNamingNotEmpty**
**Explanation:** This indicate that the naming context contains any bindings.
**User Response:** User needs to issue a separate unbind or destroy for each single binding in the sub-tree by traversing the name tree from bottom-up.

**0x49420227 CORBAMinorNamingNotFound**
**Explanation:** This indicate that the name does not identify a binding. If a compound name is passed as an argument for the bind operation, it traverses multiple contexts. A NotFound exception is raised if any of the intermediate contexts cannot be resolved.
**User Response:** Make sure the name is correct and the binding exist in the name space.

**0x49420227 CORBAMinorNamingUnknownException**
**Explanation:** An unknown exception was caught.
**User Response:** Look backward from activity log to find out the source of the problem.

**0x49420228 CORBAMinorNamingBindingIteratorObjectCreationFailed**
**Explanation:** The creation of a BindingIterator object or the narrowing to a BindingIterator implementation object failed.
**User Response:** Most likely system is running out of memory, or the remote server of this object resided is not responding.

**0x49420228 CORBAMinorNamingDCECDSCannotDeleteEntry**
**Explanation:** CDS Naming implementation cannot delete the binding because of the entry that client provided is not empty.
**User Response:** User needs to issue a separate unbind or destroy for each single binding in the sub-tree by traversing the name tree from bottom-up.

**0x49420228 CORBAMinorNamingInvalidCORBAObject**
**Explanation:** The object supplied with the invoked method is not of type CORBA::Object.
**User Response:** Make sure the object is of type CORBA::Object.

**0x49420228 CORBAMinorNamingObjectDestructionFailed**
**Explanation:** The destruction or removing of the object failed.
**User Response:** Look backward from activity log to find out the source of the problem.

**0x49420228 CORBAMinorNamingStringNameSyntaxError**
**Explanation:** The String name supplied with invoked method has a syntax error.
**User Response:** Provide a valid NamingStringSyntax String name.

**0x49420229 CORBAMinorNamingBindingStringIteratorObjectCreationFailed**
**Explanation:** The creation of a BindingStringIterator object or the narrowing to a BindingStringIterator implementation object failed.
**User Response:** Most likely system is running out of memory, or the remote server of this object resided is not responding.

**0x49420229 CORBAMinorNamingDCECDSHandleCreationFailed**
**Explanation:** CDSPI cdsGetHandle() returned NULL pointer, handle creation failed.
**User Response:** Most likely system is running out of memory or system resources.

**0x49420229 CORBAMinorNamingInvalidDCECDSName**
**Explanation:** The DCE/CDS name converted from the CORBA name supplied with the invoked method is invalid.
**User Response:** Provide a valid name for CDS-based implementation, a name n is invalid if:

1. It does not follow the constraints on CDS global names.
2. It does not confirm or translate to the ISOLatin1 character set.
3. The total name length is greater than 400.
4. The length of a simple name is greater then 254.
5. A simple name starts with the character '/'.
6. The simple name following "/.:" contains with the character '='.

**0x49420229 CORBAMinorNamingInvalidNamingContextObject**
**Explanation:** The object supplied with the invoked method is not of type CosNaming::NamingContext.
**User Response:** Make sure the object is of type CosNaming::NamingContext.

**0x4942022A CORBAMinorNamingDCECDSWrongUsageUnbind**
**Explanation:** Wrong method been called when remove a naming context (bound using bind_new_context or bind_new_context_with_string).
**User Response:** User needs to use destroy rather than unbind to remove the naming context.

**0x4942022A CORBAMinorNamingStandardSyntaxModelObjectCreationFailed**
**Explanation:** The StandardSyntaxModel create method returned a NULL object.
**User Response:** Most likely system is running out of memory or system resources.

**0x4942022B CORBAMinorNamingUtilObjectCreationFailed**
**Explanation:** The CosNameUtils Instance method returned a NULL object.
**User Response:** Most likely system is running out of memory or system resources.

**0x4942022C CORBAMinorNamingObjectCreationFailed**
**Explanation:** Failed to create or initialize object.
**User Response:** Most likely system is running out of memory or system resources, or look backward from activity log to find out the source of the problem.

**0x4942024E CORBAMinorNamingStringHeapOutOfMemory**
**Explanation:** Attempt to allocate memory on the heap has failed.
**User Response:** Most likely system is running out of memory or system resources.

**0x4942024E CORBAMinorNamingStringIllegalStringSyntax**
**Explanation:** Found a syntax error in the string.
**User Response:** Make sure follow the syntax rule for string.

**0x4942024E CORBAMinorNamingStringInvalidCORBAName**
**Explanation:** The CORBA name supplied with the invoked method is invalid.
**User Response:** A name (of type CosNaming::Name) is valid if none of the following is true:

1. It's length field is not equal to 0.
2. It does not have a component where the id of the component is a NULL pointer or the value of this id is equal to a null-string.
3. It does not have a component where the kind of the component is a NULL pointer.

**0x4942024E CORBAMinorNamingStringUnMachedQuote**
**Explanation:** Found an unmatched quote in the string.
**User Response:** Make sure a begin-quote is matched by an end-quote or vice versa.

**0x4942024F CORBAMinorNamingStringInvalidStringName**
**Explanation:** The String name supplied with the invoked method is invalid.
**User Response:** Make sure the length of String name is not zero and the String name pointer is not NULL.

**0x494202EF OTSAPPCMINOR_SPACESFOUND**
**Explanation:** Spaces have been found in a field/parameter supplied to the component. This is not supported.
**User Response:** Correct the configuration/parameter and retry. Exception is CORBA::BAD_PARAMS.

**0x494202EF OTSAPPCMINOR_UNEXPECTEDEXCEPTION**
**Explanation:** An unexpected exception has been caught by an object in the transaction service and this object is unable to process it. The exception is not a CORBA standard exception nor a CORBA user exception.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which objects were involved, were they calling an operating system function? Make sure your application has access to the resources it requires and is using any services from the operating // system or CBSeries correct. Look in the NT event logs as there may be additional messages which indicate why the exception was thrown. Exception is CORBA::UNKNOWN.

**0x494202EF SOMTRRAS::Minor_unexpectedException**
**Explanation:** An unexpected error occurred and was caught by an object in the transaction service. The error is not a CORBA standard exception nor a CORBA user exception.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which objects were involved. Where they calling an operating system function? Make sure your application has access to the resources it requires and is using any services from the operating system or Component Broker Connector correctly. Look in the NT event logs for more information which might indicate why the exception was thrown.

**0x494202F0 OTSAPPCMINOR_LONGERTHANEIGHT**
**Explanation:** A field/parameter supplied to the component is longer than its

maximum length of 8 characters. This is not supported.
**User Response:** Correct the configuration/parameter and retry. Exception is CORBA::BAD_PARAMS.

**0x494202F0 SOMTRRAS::Minor_unexpectedRetCode**
**Explanation:** An unexpected response was received by an object in the transaction service. This could be caused either by an application interacting incorrectly with the transaction service or an internal error in one of the transaction service objects.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and what point in the lifetime of the transaction the error occurred. Make sure your application is making correct use of the transaction service. Check the error log for more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x494202F1 OTSAPPCMINOR_LONGERTHANSEVENTEEN**
**Explanation:** A fully-qualified LUName field/parameter supplied to the component is longer than its maximum length of 17 characters. This is not supported.
**User Response:** Correct the configuration/parameter and retry.Exception is CORBA::BAD_PARAMS.

**0x494202F1 SOMTRRAS::Minor_unknownState**
**Explanation:** An object from the transaction service is unable to perform the action requested by an application because all, or part, of its internal state is unknown. This is a transient problem usually due to a configuration error.
**User Response:** Look for earlier messages in the activity log and follow the instructions for these messages. Exception is CORBA::INITIALIZE.

**0x494202F2 OTSAPPCMINOR_INVALIDSTATE**
**Explanation:** An object from the transaction service has detected an inconsistency in its internal state. This is caused either by an application making incorrect use of the transaction service, or there is an internal error in the transaction service.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the error occur. Make sure your application is making correct use of the transaction
// service. If your application is correct, save the NT logs and the information describing what your application was doing when the error occurred and contact your support organization. Exception is CORBA::INTERNAL, CosTransactions::NotPrepared.

**0x494202F2 OTSAPPCMINOR_LONGERTHANSIXTYFOUR**
**Explanation:** A Transaction Program Name (TPN) field/parameter supplied to the component is longer than its maximum length of 64 characters. This is not supported.
**User Response:** Correct the configuration/parameter and retry. Exception is CORBA::BAD_PARAMS.

**0x494202F2 SOMTRRAS::Minor_invalidState**
**Explanation:** An object from the transaction service has detected an inconsistency in its internal state. This is caused either by an application making incorrect use of the transaction service, or there is an internal error in the transaction service.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and what point in the lifetime of the transaction the error occurred. Make sure your application is making correct use of the transaction service. Check the error log for

more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x494202F3 OTSAPPCMINOR_ZEROLENGTH**
**Explanation:** A field/parameter supplied to the component is either NULL or zero length. This is not supported.
**User Response:**Correct the configuration/parameter and retry. Exception is CORBA::BAD_PARAMS.

**0x494202F3 SOMTRRAS::Minor_unfinishedTransaction**
**Explanation:** The transaction service is unable to commit a top-level transaction because there is still outstanding work occurring for the transaction either locally or in a remote server. Alternatively, a non-user exception was generated in a remote method.
**User Response:** Look in the NT event logs to see if an exception occurred in a remote server. If a non-user exception is being thrown, correct your application so that the exception does not occur, or ensure the exception is caught in the server where it is first thrown. If no exception has occurred, there is a problem in the design of your application. It should not request a commit until all remote methods and subtransactions have completed for the transaction. Correct your application.

**0x494202F4 OTSAPPCMINOR_TOOMANYDOTS**
**Explanation:** A fully qualified LU name field/parameter supplied to the component contains more then one ".". This is not supported.
**User Response:** Correct the configuration/parameter and retry. Exception is CORBA::BAD_PARAMS.

**0x494202F4 SOMTRRAS::Minor_unfinishedSubTransaction**
**Explanation:** The transaction service is unable to commit a subtransaction because there is either still outstanding work occurring for the transaction or a non-user exception was thrown and not caught in a remote method.
**User Response:** Look in the NT event logs to see if an exception occurred in a remote server involved in the transaction. If a non-user exception is being thrown, correct your application so that the exception does not occur, or ensure the exception is caught in the server where it is first thrown. If no exception has occurred, there is a problem in the design of your application. It should not request a commit until all remote methods and subtransactions have completed for the transaction. Correct your application.

**0x494202F5 OTSAPPCMINOR_BUFFERTOOSMALL**
**Explanation:** A buffer passed to an internal routine is too small.
**User Response:** Turn on transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::INTERNAL.

**0x494202F5 SOMTRRAS::Minor_noMemory**
**Explanation:** An object from the transaction service is unable to acquire the memory it needs to complete a request from an application.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and at what point in the lifetime of the transaction the error occurred. Make sure your application is making correct use of the transaction service. In particular look for possible loops that may consume memory. If the application runs for some time before this exception occurs, ensure that objects which are no longer required are being correctly destroyed. If your application is correct, increase your NT virtual page size.

**0x494202F6 OTSAPPCMINOR_BADLOGNAME**
**Explanation:** A log name received from a partner system is not specified in the correct SNA format.
**User Response:**Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202F6 OTSAPPCMINOR_LOGICERROR**
**Explanation:** The transaction service has detected an internal logic error. This should not occur if the transaction service is being called correctly.
**User Response:**Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the error occur. Make sure your application is making correct use of the transaction
// service. If your application is correct, save the NT logs and the information describing what your application was doing when the error occurred and contact your support organization. Exception is CORBA::INTERNAL.

**0x494202F6 SOMTRRAS::Minor_logicError**
**Explanation:** The transaction service has detected an internal logic error. This should not occur if the transaction service is being called correctly.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and what point in the lifetime of the transaction the error occurred. Make sure your application is making correct use of the transaction service. Check the error log for more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x494202F7 OTSAPPCMINOR_BADPSHEADER**
**Explanation:** A presentation services (PS) header received from a partner system during commit processing is not specified in the correct SNA format.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202F7 OTSAPPCMINOR_USERERROR**
**Explanation:** The transaction service has been incorrectly called by an application.
**User Response:**Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the exception occur. Look for other messages in the NT event logs that may give more details on the cause of the problem. When you have identified which call or calls to the transaction service are in error, correct your application. Exception is CORBA::INTERNAL.

**0x494202F7 SOMTRRAS::Minor_userError**
**Explanation:** The transaction service was incorrectly called by an application.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the exception occur. Look for other messages in the NT event logs that may give more details on the cause of the problem. When you have identified which call or calls to the transaction service are in error, correct your application.

**0x494202F8 OTSAPPCMINOR_BADLUWID**
**Explanation:** A Logical Unit of Work identifier (LUWId) received from a partner system is not specified in the correct SNA format.

**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202F8 SOMTRRAS::Minor_mutexError**
**Explanation:** The transaction service was unable to lock or unlock a mutex. This should not occur if the transaction service is being called correctly.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the error occur. Make sure your application is making correct use of the transaction service. Check the error log for more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x494202F9 OTSAPPCMINOR_BADLUNAME**
**Explanation:** A Logical Unit (LU) name received from a partner system is not specified in the correct SNA format.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x49202F9 SOMTRRAS::Minor_noTransaction**
**Explanation:** A transactional request was issued outside the scope of the transaction. This is an application error.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the exception occur. Ensure that your application uses the CosTransactions::Current interface to ensure a transaction context is associated with the current thread of execution when the transaction service is called.

**0x494202FA OTSAPPCMINOR_BADSESSIONID**
**Explanation:** A session identifier received from a partner system is not specified in the correct SNA format.
**User Response:**Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202FA SOMTRRAS::Minor_wrongTransaction**
**Explanation:** A transactional request was issued in the scope of one transaction and returned in the scope of another transaction. This is an application error.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and at what point in the lifetime of the transaction the exception occurred. In particular, identify the objects that are located in different servers to where the transaction was started and that either create one or more subtransactions or use the CosTransactions::Current interface to suspend or resume the transaction context. It is likely that these objects are not managing transactions correctly.

**0x494202FB OTSAPPCMINOR_BADCONVCORRELATOR**
**Explanation:** A conversation correlator received from a partner system is not specified in the correct SNA format.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x49202FB SOMTRRAS::Minor_retryLimitExhausted**
**Explanation:** There is a problem communicating with the objects involved in a transaction. The transaction service made as many attempts to contact this object as are permitted by the commitRetryLimit configured for the server and made a heuristic decision.
**User Response:** Restart any servers that are currently unavailable. This completes the transaction for any registered objects that are located in these servers. Look at the action that was taken by each of these objects and correct any problems in the data for your application caused by this heuristic decision.

**0x494202FC OTSAPPCMINOR_BADLUWSTATE**
**Explanation:** A transaction state indicator received from a partner system is not a recognized SNA value.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202FC SOMTRRAS::Minor_wrongState**
**Explanation:** The transaction service is unable to complete the requested operation because it is not in the correct state to perform the work. This is an application error.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and at what point in the lifetime of the transaction the exception occurred. It is likely that these objects are not managing transactions correctly. Correct your application as appropriate.

**0x494202FD OTSAPPCMINOR_BADXLN**
**Explanation:** An exchange log names (XLN) record received from a partner system is not
// in the correct SNA format.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202FD SOMTRRAS::Minor_timedOut**
**Explanation:** A transaction rolled back because it did not complete within the specified time limit. This may indicate that the server is waiting for a resource that is unavailable, or a deadlock has occurred in your application.
**User Response:** Check that all of the servers involved in the transaction are operating correctly. In addition, check the use of locks in your application for possible deadlock situations.

**0x494202FE OTSAPPCMINOR_BADCOMPARESTATES**
**Explanation:** An compare states record received from a partner system is not in the correct SNA format.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exception is CORBA::MARSHAL.

**0x494202FE SOMTRRAS::Minor_notInitialized**
**Explanation:** An object from the transaction service was requested to perform an operation before it was initialized.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and at what point in the lifetime of the transaction the error occurred. Make sure

your application is making correct use of the transaction service. If your application is correct, save the NT logs and the information describing what your application was doing when the error occurred and report the occurrence to technical support.

**0x494202FF OTSAPPCMINOR_NULLOBJECT**
**Explanation:** A transaction service request has been made against an object that does not exist.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which transaction service objects were involved and what point in the lifetime of the transaction did the error occur. Make sure your application is making correct use of the transaction service. If your application is correct, save the NT logs and the information describing what your application was doing when the error occurred and contact your support organization. Exceptions are CORBA::PERSIST_STORE, CORBA::INTERNAL and CORBA::INVALID_TRANSACTION, CORBA::NO_MEMORY.

**0x494202FF OTSAPPCMINOR_SENDRECEIVEFAILED**
**Explanation:** A call to send data from a partner system and receive a response has failed.
**User Response:** Turn on link trace in the local SNA product and transaction service trace in the Component Broker server and rerun the failing request. Then contact IBM Service. Exceptions are OTSAPPCConnection::TPNUnavailable, OTSAPPCConnection::SecurityNotValid, OTSAPPCConnection::BadCommunications, OTSAPPCConnection::PartnerNotRecoverable, and OTSAPPCConnection::RecoveryIncomplete.

**0x494202FF SOMTRRAS::Minor_nullObject**
**Explanation:** A transaction service request was made against an object that does not exist.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which transaction service objects were involved and at what point in the lifetime of the transaction the error occurred. Make sure your application is making correct use of the transaction service. Check the error log for more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x49420300 OTSAPPCMINOR_COMMITCOMMFAILURE**
**Explanation:** A commit call is unable to complete because a partner system is unreachable.
**User Response:** Restart the partner system. Details of which system is unavailable is given in a previous message. Exception is CORBA::COMM_FAILURE.

**0x49420300 SOMTRRAS::Minor_recoveryFailed**
**Explanation:** The transaction service has experienced problems during recovery and is unable to accept new requests.
**User Response:** If transactions are not used in this server then no action is needed. If transactions are required, look for the messages logged just prior to this exception being logged as these explain why the transaction service is unavailable.

**0x49420301 OTSAPPCMINOR_RBCOMMFAILURE**
**Explanation:** A rollback call is unable to complete because a partner system is unreachable.
**User Response:** Restart the partner system. Details of which system is unavailable is given in a previous message. Exception is CORBA::COMM_FAILURE.

**0x49420301 SOMTRRAS::Minor_terminationNotAllowed**
**Explanation:** An application has attempted to end a transaction using the commit method in a different server from the one where the transaction was started. This is not supported by the transaction service.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, look at which application objects were involved and at what point in their processing the commit method call was made. Correct your application so that the commit is called in the same server as the begin method. The easiest way to ensure this is to have a single object responsible for calling both the begin and the commit method.

**0x49420302 OTSAPPCMINOR_RESYNCOK**
**Explanation:** A resync request confirms that the transaction has already completed on both systems.
**User Response:** No action is required. This exception is thrown for information only. Exception is CosTransactions::NoTransaction.

**0x49420302 SOMTRRAS::Minor_rollbackOnlySet**
**Explanation:** The transaction was marked "rollback only" and one of the following types of requests was attempted.

- Start a subtransaction
- Register a CosTransactions::Resource object with a CosTransactions::Coordinator for the transaction
- Register a CosTransactions::Synchronization object with a CosTransactions::Coordinator for the transaction
- Make a remote request
- Receive a remote request

These operations are not permitted when the transaction is marked as "rollback only".
**User Response:** Examine the design of your application to understand how the transaction service is used. Either alter the code so that these calls are not made when the transaction is marked "rollback only", or ensure your application uses a try/catch structure around these calls so it can handle the exception.

**0x49420303 OTSAPPCMINOR_PREPAREHEURISTIC**
**Explanation:** A heuristic decision was detected during a prepare operation. This heuristic decision has been made by one of your application's CosTransactions::Resource objects. The transaction service is reporting the heuristic decision as it will affect the outcome of the transaction.
**User Response:** None. Exceptions are CosTransactions::HeuristicMixed, CosTransactions::HeuristicHazard, CosTransactions::HeuristicCommit, and CosTransactions::HeuristicRollback.

**0x4920303 SOMTRRAS::Minor_prepareHeuristic**
**Explanation:** A heuristic decision was detected during a prepare operation. This heuristic decision was made by one of the CosTransactions::Resource objects of your application. The transaction service is reporting the heuristic decision as it affects the outcome of the transaction.
**User Response:** None.

**0x49420304 OTSAPPCMINOR_COMMITHEURISTIC**
**Explanation:** A heuristic decision was detected during a commit operation. This heuristic decision has been made by one of your application's CosTransactions::Resource objects. The transaction service is reporting the

heuristic decision as it will affect the outcome of the transaction.
**User Response:**None. Exceptions are CosTransactions::HeuristicMixed, CosTransactions::HeuristicHazard, and CosTransactions::HeuristicRollback.

**0x4920304 SOMTRRAS::Minor_commitHeuristic**
**Explanation:** A heuristic decision was detected during a commit operation. This heuristic decision was made by one of the CosTransactions::Resource objects of your application. The transaction service is reporting the heuristic decision as it affects the outcome of the transaction.
**User Response:** None.

**0x49420305 OTSAPPCMINOR_ROLLBACKHEURISTIC**
**Explanation:** A heuristic decision was detected during a rollback operation. This heuristic decision has been made by one of your application's CosTransactions::Resource objects. The transaction service is reporting the heuristic decision as it will affect the outcome of the transaction.
**User Response:** None. Exceptions are CosTransactions::HeuristicMixed, CosTransactions::HeuristicHazard, and CosTransactions::HeuristicCommit.

**0x4920305 SOMTRRAS::Minor_rollbackHeuristic**
**Explanation:** A heuristic decision was detected during a rollback operation. This heuristic decision was made by one of the CosTransactions::Resource objects of your application. The transaction service is reporting the heuristic decision as it affects the outcome of the transaction.
**User Response:** None.

**0x49420306 SOMTRRAS::Minor_subtransactionRolledBack**
**Explanation:** A subtransaction rolled back during a commit_subtransaction call to a CosTransactions::Resource object. This exception is part of the standard transaction service interface.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to handle this exception appropriately.

**0x49420307 OTSAPPCMINOR_COMMIT1PROLLEDBACK**
**Explanation:** A transaction rolled back during the commit_one_phase call to a CosTransactions::Resource object. This exception is part of the standard transaction service interface.
**User Response:**Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to handle this exception appropriately. Exception is CORBA::TRANSACTION_ROLLEDBACK.

**0x49420307 SOMTRRAS::Minor_commitOnePhaseRolledBack**
**Explanation:** A transaction rolled back during the commit_one_phase call to a CosTransactions::Resource object. This exception is part of the standard transaction service interface.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to handle this exception appropriately.

**0x49420308 OTSAPPCMINOR_COMMIT1PHEURISTIC**
**Explanation:** A heuristic decision was detected during a commit_one_phase call to a CosTransactions::Resource object. This heuristic decision has been made by one of your application's CosTransactions::Resource objects. The transaction service is

reporting the heuristic decision as it will affect the outcome of the transaction.
**User Response:** None. Exceptions are: CosTransactions::HeuristicMixed and CosTransactions::HeuristicHazard.

**0x4920308 SOMTRRAS::Minor_commitOnePhaseHeuristic**
**Explanation:** A heuristic decision was detected during a commit_one_phase call to a CosTransactions::Resource object. This heuristic decision was made by one of the CosTransactions::Resource objects of your application. The transaction service is reporting the heuristic decision as it affects the outcome of the transaction.
**User Response:** None.

**0x49420309 SOMTRRAS::Minor_notSubtransaction**
**Explanation:** A method that should only be used when working on behalf of a subtransaction was issued during a top-level transaction. This is an application error.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to call this object appropriately.

**0x4942030A SOMTRRAS::Minor_notTopLevelTransaction**
**Explanation:** A method that should only be used when working on behalf of a top-level transaction was issued during a subtransaction. This is an application error.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to call this object appropriately.

**0x4942030B SOMTRRAS::Minor_commitRolledBack**
**Explanation:** A transaction rolled back during the commit call. This exception is part of the standard transaction service interface.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to handle this exception appropriately.

**0x4942030C SOMTRRAS::Minor_noFactoryAvailable**
**Explanation:** A transaction could not be started because a CosTransactions::TransactionFactory is not available.
**User Response:** Ensure there is an application server available which contains a transaction factory for the client.

**0x4942030D SOMTRRAS::Minor_activeNested**
**Explanation:** A transaction could not be completed because there are active subtransactions. This is an application error.
**User Response:** Correct your application.

**0x4942030E OTSAPPCMINOR_NOTSUPPORTED**
**Explanation:** The request could not be completed because the function is not supported.
**User Response:** Identify which call to the transaction service is throwing this exception. Check the documentation for this call and alter your application to use supported operations. Exceptions are CORBA::NO_IMPLEMENT.

**0x492030E SOMTRRAS::Minor_notSupported**
**Explanation:** The request could not be completed because the function is not supported.

**User Response:** Identify which call to the transaction service is throwing this exception. Check the documentation for this call and alter your application to use supported operations.

**0x4942030F SOMTRRAS::Minor_omgtidAlreadyUsed**
**Explanation:** The transaction service can not create a new transaction because it is unable to generate a new OMGtid as all possible values for the OMGtid are in use. This could be caused by an application starting many transactions and never ending them.
**User Response:** Investigate what your application was doing when the exception was thrown. For example, which objects were involved and what where they doing. Look at the points in your application where transactions are started and ended and ensure your application is making correct use of the transaction service. If your application is correct, save the NT logs and the information describing what your application was doing when the error occurred and contact your support organization. Exception is CORBA::IMP_LIMIT.

**0x49420310 SOMTRRAS::Minor_omgtidNull**
**Explanation:** The CosTransactions::TransactionFactory recreate method was called with a propagation context parameter containing a NULL OMGtid. This method is provided for use by implementations of the transaction service when receiving transactional requests from remote servers. It should not be used by applications.
**User Response:** Investigate what your application was doing when the exception was thrown. Look at which objects were involved and what they were doing. Ensure that no part of your application is calling the CosTransactions::TransactionFactory recreate method. If the error occurs while your application is not using the recreate method then the transaction service is using this method when receiving a request from a remote server. Look for errors in this remote server. If you cannot solve the problem, save the NT logs and the information describing what your application was doing when the error occurred and report the occurrence to technical support.

**0x49420311 SOMTRRAS::Minor_alreadyCurrent**
**Explanation:** The transaction service cannot receive a transactional request for a local object because the thread where the request is to run is already associated with this transaction. This is an internal error in the transaction service.
**User Response:** Check the error log for more information that might describe what your application was doing when the error occurred and report the occurrence to technical support.

**0x49420312 SOMTRRAS::Minor_noLog**
**Explanation:** The transaction service can not initialize properly because its log file is not available. This may be because the log directory set up for the server does not exist, there is insufficient disk space for the log files, or the server does not have permission to create the log files in the log directory.
**User Response:** Restart your server and look for the messages generated by the transaction service. These explain what is wrong with the log configuration. Correct the log configuration and restart the server.

**0x49420313 OTSAPPCMINOR_RESYNCHEURISTIC**
**Explanation:** A heuristic decision was taken when the transaction service was resolving the outcome of an incomplete transaction during server startup. This heuristic decision has been made by one of your application's CosTransactions::Resource objects. The transaction service is reporting the heuristic decision as it will affect the outcome of the transaction.

**User Response:** None. Exceptions are CosTransactions::HeuristicMixed, CosTransactions::HeuristicHazard, CosTransactions::HeuristicCommit, and CosTransactions::HeuristicRollback.

**0x4920313 SOMTRRAS::Minor_resyncHeuristic**
**Explanation:** A heuristic decision was taken when the transaction service was resolving the outcome of an incomplete transaction during server startup. This heuristic decision was made by one of the CosTransactions::Resource objects of your application. The transaction service is reporting the heuristic decision as it affects the outcome of the transaction.
**User Response:** None.

**0x49420314 SOMTRRAS::Minor_asyncRolledBack**
**Explanation:** A request can not be performed on a transaction because this transaction has already rolled back. This exception is part of the standard transaction service interface.
**User Response:** Identify which call to the transaction service resulted in this exception. Check the documentation for this call and ensure your application is programmed to handle this exception appropriately.

**0x49420315 SOMTRRAS::Minor_thisServerEnding**
**Explanation:** The transaction is being rolled back because the local server is terminating.
**User Response:** Rerun the transaction when the server is restarted.

**0x49420316 SOMTRRAS::Minor_remoteServerEnding**
**Explanation:** The transaction is being rolled back because another server which is processing part of the transaction is terminating.
**User Response:** Rerun the transaction when the remote server is restarted.

**0x49420317 SOMTRRAS::Minor_superiorRegFailed**
**Explanation:** The transaction is being rolled back because another server which is processing part of the transaction can not be contacted.
**User Response:** Rerun the transaction when the remote server is restarted.

**0x49420318 SOMTRRAS::Minor_noLogStorage**
**Explanation:** The transaction service is unable to continue running transactions because there is insufficient disk space for the transaction service log. This log contains information about the transactions running in the server and is used during recovery.
**User Response:** Free some disk space for the transaction service log.

**0x49420320 OTSAPPCMINOR_ROLLBACKONLYSET**
**Explanation:** The transaction has been marked "rollback only" and one of the following types of requests has been attempted:

- start a new APPC connection
- update a database for the first time during the transaction
- register a CosTransactions::Resource object with a CosTransactions::Coordinator for the transaction
- register a CosTransactions::Synchronization object with a CosTransactions::Coordinator for the transaction
- make a remote request
- receive a remote request

These operations are not permitted when the transaction is marked as "rollback only".
**User Response:** Examine the design of your application to understand how the transaction service is used. Either alter the code so that these calls are not made when the transaction is marked "rollback only", or ensure your application uses a try/catch structure around these calls so it can handle the CORBA::TRANSACTION_ROLLEDBACK exception. Exception is CORBA::TRANSACTION_ROLLEDBACK.

**0x494203EC SOMIM_INTERNAL**
**Explanation:** Unknown.
**User Response:** Report the occurrence to technical support.

**0x4942041A SOMROOTIM_ICONTAINERMINOR_HEAP_OUT_OF_MEMORY**
**Explanation:** Memory could not be allocated from a heap.
**User Response:** A heap could not extend during an allocation. Review the logs to determine the heap and take action to allow it to extend.

**0x4942041A SOMROOTIM_ICONTAINERMINOR_NO_LOCALFRIENDQOS**
**Explanation:** An object cannot register with a container.
**User Response:** The object lacks the interface needed to register with a container. Refer to the programming documentation to determine the interface needed.

**0x4942041A SOMROOTIM_ICONTAINERMINOR_NOIMPLEMENT**
**Explanation:** The invoked operation is not supported in the product or is not valid on the target object.
**User Response:** Check that the operation being invoked and the runtime type of the target object are compatible. Reference the documentation for the operation to find out about any restrictions.

**0x4942041A SOMROOTIM_ICONTAINERMINOR_UNEXPECTED**
**Explanation:** An unexpected error occurred during an operation.
**User Response:** This indicates a system type failure. Contact your service provider for additional help.

**0x4942041B SOMROOTIM_ICONTAINER_MC_FAILED_REGISTRATION**
**Explanation:** The master container did not register into the master container object cache.
**User Response:** The master container failed to register with itself. Restart the server, if this fails again contact your service provider.

**0x4942041B SOMROOTIM_ICONTAINERMINOR_BAD_KEY**
**Explanation:** The key provided is invalid or corrupted.
**User Response:** Retry the operation, if the problem persists review the objects key for valid character set and syntax. If this is not the problem contact your service provider.

**0x4942041B SOMROOTIM_ICONTAINERMINOR_UNKNOWN_HEAP**
**Explanation:** An unexpected error occurred during the use of a heap.
**User Response:** Review the logs to determine the exact cause of this failure and take the recommended action.

**0x4942044C SOMBOIM_IHOMEMINOR_BADPARAM_CONFIG**
**Explanation:** Configuration of the home failed.
**User Response:** Check the associated message. Most often this is the result of an

incorrect DULL or function name. Update the COS with the correct value, or rebuild the particular DULL to have the function specified and try again.

**0x4942044C SOMBOIM_IHOMEMINOR_DUPLICATE_KEY**
**Explanation:** The key given for a createFromPrimaryKey has already been used by an existing object.
**User Response:** Perform a findByPrimaryKeyString to find objects which already exist.

**0x4942044C SOMBOIM_IHOMEMINOR_INVALID_KEY**
**Explanation:** The key given for a createFromPrimaryKeyString or findByPrimaryKeyString can not be used to create or reactivate the object.
**User Response:** Check the contents of the key to make sure they are correct. Check to make sure the type of key used is compatible with the keys used by the home in question.

**0x4942044C SOMBOIM_IHOMEMINOR_NO_IMPL**
**Explanation:** The invoked operation is not supported in the product or is not valid on the target object.
**User Response:** Check that the operation being invoked and the runtime type of the target object are compatible. Reference the documentation for the operation to find out about any restrictions.

**0x4942044C SOMBOIM_IHOMEMINOR_NO_MEMORY_CONFIG**
**Explanation:** A memory error occurred during configuration of a home. Memory could not be extended in the heap.
**User Response:** Review the log and check to see why the heap could not be used to create the object.

**0x4942044C SOMBOIM_IHOMEMINOR_UNINIT_FAILED**
**Explanation:** The uninitForPassivation or uninitForDestruction method failed to complete.
**User Response:** In the case of uninitForDestruction, check to make sure that the state of the home is "dead". If not, the home can not be destroyed. Otherwise, check the error log for previous messages which may indicate why the the uninit method failed.

**0x4942044C SOMBOIM_IHOMEMINOR_UNKNOWN_CONFIG**
**Explanation:** An error during the configuration has occurred.
**User Response:** This is a global configuration exception. Check previous error log entries for previous configuration exceptions. If these problems are fixed, this exception should cease to occur as well.

**0x4942044C SOMBOIMHOME_IHOMEMINOR_SMOCOHOMENARROW**
**Explanation:** The SMOCO Home could not be narrowed to a home of the correct type.
**User Response:** Make sure your CDS default application was built correctly.

**0x4942044D SOMBOIM_IHOMEMINOR_BADPARAM_PO**
**Explanation:** Contents of the key passed in invalid.
**User Response:** Check the contents of the key to determine if this is the correct key for this particular object.

**0x4942044D SOMBOIM_IHOMEMINOR_NO_MEMORY_CREATE_MO**
**Explanation:** A memory error occurred during construction of a managed object.

Memory could not be extended in the heap.
**User Response:** Review the log and check to see why the heap could not be used to create the object.

**0x4942044D SOMBOIMHOME_IHOMEMINOR_SMOCONARROW**
**Explanation:** The SMOCO could not be narrowed to the correct type.
**User Response:** Make sure your CDS default application was built correctly.

**0x4942044D SOMBOIM_IHOMEMINOR_UNKNOWN_CTOR**
**Explanation:** An unexpected error occurred during the construction of a home. This error means that the home caught an exception during its creation logic, and has thrown this exception.
**User Response:** Check previous log entries to determine the nature of the exception(s) prior to this exception.

**0x4942044E SOMBOIM_IHOMEMINOR_BADPARAM_SYNCFROM**
**Explanation:** The home could not retrieve all its data from the CDS or the home could not configure itself from the data given.
**User Response:** Check the previous exception in the error log to see if a configuration error occurred. If so, take action to correct the configuration value. If a configuration error did not occur, check the previous entries in the log to see if the CDS Data Object could not find a particular value in the CDS. If so, correct the CDS.

**0x4942044E SOMBOIM_IHOMEMINOR_COMPLETEFAILED**
**Explanation:** The SMOCO and its home could not be revived.
**User Response:** Make sure your CDS default application was built correctly.

**0x4942044E SOMBOIM_IHOMEMINOR_NO_MEMORY_CREATE_DO**
**Explanation:** A memory error occurred during construction of a data object. Memory could not be extended in the heap.
**User Response:** Review the log and check to see why the heap could not be used to create the object.

**0x4942044E SOMBOIM_IHOMEMINOR_UNKNOWN_DTOR**
**Explanation:** An unexpected error occurred during the destruction of a home. This error means that the home caught an exception during its destruction logic, and has thrown this exception.
**User Response:** Check previous log entries to determine the nature of the exception(s) prior to this exception.

**0x4942044F SOMBOIM_IHOMEMINOR_BADPARAM_MIXIN**
**Explanation:** The mixin type used can not be used by the home.
**User Response:** Check the type of the mixin.

**0x4942044F SOMBOIM_IHOMEMINOR_NO_MEMORY_CREATE_PK**
**Explanation:** A memory error occurred during construction of a primary key. Memory could not be extended in the heap.
**User Response:** Review the log and check to see why the heap could not be used to create the object.

**0x4942044F SOMBOIM_IHOMEMINOR_UNKNOWN_CREATE_PK**
**Explanation:** An unknown error occurred curing the creation of a primary key
**User Response:** Check the logic of the keys constructor to see that is behaving properly.

**0x49420450 SOMBOIM_IHOMEMINOR_BAD_KEY**
**Explanation:** Key component could not be created.
**User Response:** An internal error has occurred. Check for memory heap overflows.

**0x49420450 SOMBOIM_IHOMEMINOR_UNKNOWN_CREATE_FROM_PK**
**Explanation:** During a creation, find, or reactivation, an unknown error occurred.
**User Response:** This is a catcher exception. To find the cause of the problem, examine the previous error log entries.

**0x49420451 SOMBOIM_IHOMEMINOR_MCINIT**
**Explanation:** The master container or bootstrap home failed to initialize.
**User Response:** Make sure your CDS default application was built correctly.

**0x49420451 SOMBOIM_IHOMEMINOR_NULL_DO_PARAM**
**Explanation:** A null DO was passed into the home during an initForReactivation call.
**User Response:** An internal error has occurred.

**0x49420451 SOMBOIM_IHOMEMINOR_UNKNOWN_CREATE_DO**
**Explanation:** An unknown error occurred curing the creation of a data object.
**User Response:** Check the logic of the constructor for the data object to see that is behaving properly.

**0x49420452 SOMBOIM_IHOMEMINOR_PSTATE**
**Explanation:** An error occurred as the result of the home processing its state.
**User Response:** Check the associated message to see what has occurred. Most often, this is the result of an object changing its state.

**0x49420452 SOMBOIM_IHOMEMINOR_STATE**
**Explanation:** An error occurred as the result of the home processing its state.
**User Response:** Check to see that the home has a valid state attribute in the CDS.

**0x49420453 SOMBOIM_IHOMEMINOR_BAD_KC**
**Explanation:** An error occurred processing the homeIdentification method on the home.
**User Response:** Check the value of the managedObjectClass for the home and make sure that it is valid.

**0x49420453 SOMBOIM_IHOMEMINOR_UNKNOWN_CREATE_MO**
**Explanation:** An unknown error occurred during the creation of a managed object.
**User Response:** Check the logic of the constructor for the managed object to see that is behaving properly.

**0x49420454 SOMBOIM_IHOMEMINOR_UNKNOWN_DELETE_PK**
**Explanation:** An unknown error occurred during the destruction of a primary key.
**User Response:** Check the logic of the primary keys' destructor to see that is behaving properly.

**0x49420455 SOMBOIM_IHOMEMINOR_CREATE_DO**
**Explanation:** An unknown error occurred during the creation of a data object.
**User Response:** Check the logic of the constructor for the data object to see that is behaving properly.

**0x49420456 SOMBOIM_ICDSMINOR_IMKEYCOLLECTION_NOT_SET**
**Explanation:** The key collection was not set in the CDS data object.

**0x49420456 SOMBOIM_ICDSMINOR_INITDO_CDS**
**Explanation:** A exception was thrown by the default data store.
**User Response:** See message log for more information.

**0x49420456 SOMBOIM_ICDSMINOR_UNKNOWN**
**Explanation:** An unexpected error occurred during an operation.
**User Response:** See message log for more details.

**0x49420456 SOMBOIM_SMOCOMINOR_INTERNAL_FAILURE**
**Explanation:** An internal failure occurred in the SMOCO. In most situations this is caused by an incorrect CDS image.
**User Response:** Check the error log for more information.

**0x49420457 SOMBOIM_ICDSMINOR_INITDO**
**Explanation:** An unknown exception occurred while accessing the default data store.
**User Response:** See message log for more information.

**0x49420457 SOMBOIM_ICDSMINOR_REF_ELEMENT**
**Explanation:** An error occurred while attempting to get a reference by name.
**User Response:** See message log for more details.

**0x49420458 SOMBOIM_ICDSMINOR_DELETE**
**Explanation:** A delete of a CDS image did not complete, since the CDS image was in the wrong state.

**0x49420459 SOMBOIM_ICDSMINOR_GET_SET_ATTRIBUTE_FAILED**
**Explanation:** An error occurred while using an attribute in the CDS.
**User Response:** See the message log for more details.

**0x4942045A SOMBOIM_ICDSMINOR_INVALID_INDEX**
**Explanation:** An invalid index was requested.

**0x49420460 SOMBOIM_ICONTAINERMINOR_DEAD_STATE_CDS**
**Explanation:** A container has reactivated in a terminating state.
**User Response:** The container is terminating and its image are deleted from CDS. There is no action for this exception, it is informational only.

**0x49420460 SOMBOIM_ICONTAINERMINOR_HEAP_OUT_OF_MEMORY**
**Explanation:** Memory could not be allocated from a heap.
**User Response:** A heap could not extend during an allocation. Review the logs to determine the heap and take action to allow it to extend.

**0x49420460 SOMBOIM_ICONTAINERMINOR_NO_LOCALFRIENDQOS**
**Explanation:** An object cannot register with a container.
**User Response:** The object lacks the interface needed to register with a container. Refer to the programming documentation to determine the interface needed.

**0x49420460 SOMBOIM_ICONTAINERMINOR_NOIMPLEMENT**
**Explanation:** The invoked operation is not supported in the product or is not valid on the target object.
**User Response:** Check that the operation being invoked and the runtime type of the target object are compatible. Reference the documentation for the operation to find out about any restrictions.

**0x49420461 SOMBOIM_ICONTAINERMINOR_UNEXPECTED**
**Explanation:** An unexpected error occurred during an operation.
**User Response:** This indicates a system type failure. Contact your service provider for additional help.

**0x49420461 SOMBOIM_ICONTAINERMINOR_UNKNOWN_HEAP**
**Explanation:** An unexpected error occurred during the use of a heap.
**User Response:** Review the boimLog and the rootLog to determine the exact cause of this failure and take the recommended action.

**0x49420461 SOMBOIM_ICONTAINERMINOR_UNKNOWN_SETREFDATA**
**Explanation:** The reference data received from the ORB is invalid.
**User Response:** Retry the operation, if the problem persists review the objects key for valid character set and syntax. If this is not the problem contact your service provider.

**0x49420462 SOMBOIM_ICONTAINERMINOR_BAD_KEY**
**Explanation:** The key provided is invalid or corrupted.
**User Response:** Retry the operation, if the problem persists review the objects key for valid character set and syntax. If this is not the problem contact your service provider.

**0x49420463 SOMBOIM_ICONTAINERMINOR_NO_MORE_KEYS**
**Explanation:** The provided key did not resolve an object.
**User Response:** The key has become corrupted. Resolve a new object reference.

**0x49420464 SOMBOIM_ICONTAINERMINOR_REACTIVATION_FAILED**
**Explanation:** The object is not in memory and cannot be placed in memory.
**User Response:** This is an internal failure. Gather information from the boimLog and rootLog and contact your service provider.

**0x49420465 SOMBOIM_ICONTAINERMINOR_NO_ACTION_NECESSARY**
**Explanation:** An internal exception has escaped to the user interface.
**User Response:** None, ignore this error.

**0x49420466**
**SOMBOIM_ICONTAINERMINOR_CAN_NOT_HOLD_THIS_TRANSIENT**
**Explanation:** A transient object is being resolved to an object that does not exist in the container.
**User Response:** Create the object into the container before resolving it. A transient object cannot be reactivated, it must be created before being referenced.

**0x49420467 SOMBOIM_ICONTAINERMINOR_NO_HOMES_DEFINED**
**Explanation:** A home cannot be found to create the requested object.
**User Response:** The container does not have a home capable of creating the requested object. Review the containers backing store (CDS) to assure the objects home is configured on this container. Review the logs (boimLog and rootLog) to assure the creation of the home did not fail during container initialization.

**Minor Code Ranges**
All minor error codes in Component Broker are defined to fall within a specific numeric range. This range is defined by the Object Management Group. Within this range, Component Broker defines further ranges. Each Component Broker range corresponds to a specific Component Broker component. This makes it possible to identify which component threw an exception by reading the minor error code.

While it is not necessary for developers to know which component threw an exception to identify an error, it can be helpful.

Minor codes are scope to System Exceptions in the range from 0 to 4095 (hexadecimal 000 to FFF). A minor code must be a unique number within the scope for each System Exception, but there is no restriction that minor codes be unique across all System Exceptions.

**Table 5. Code Ranges by Component**

| Code Range (Hexadecimal) | Component or Service |
|---|---|
| 0000-0031 | Common error codes. For additional details, see the list following this table (page 259). |
| 0032-0063 | Object Request Broker (ORB) |
| 0064-0095 | Java client |
| 0096-00C7 | Server run time |
| 00C8-00F9 | Server Groups |
| 00FA-012B | Caching |
| 012C-015D | IOM |
| 015E-018F | Concurrency service |
| 0190-01C1 | Events service |
| 01C2-01F3 | Externalization service |
| 01F4-0225 | LifeCycle service |
| 0226-0257 | Naming service |
| 0258-0289 | Object Identity service |
| 028A-02BB | Query service |
| 02BC-02ED | Security service |
| 02EE-031F | Transaction service |
| 0320-03E7 | Reserved |
| 03E8-0419 | Application Adapter Framework |
| 041A-044B | Root Application Adapter |
| 044C-047D | Business Object Application Adapter |
| 047E-04AF | Relational DB Application Adapter |
| 04B0-0FFF | Reserved |

Usually, components raise exceptions using minor codes (from their assigned range). However, a set of common error codes are provided for use by all Component Broker components. These minor codes are defined identifiers that are used to set minor code fields when returning exceptions. These messages are also defined in the RAS message catalog. These messages are used when storing errors in the logs.

### Common Error Codes

The following is a list of the common error codes:

**Code (Hexadecimal): 0000**
**Description:** Minor code is unspecified. No special meaning.
**Minor Code:** CORBAMinorCommonUnspecified
**Message ID:** CommonMessageUnspecified

**Code (Hexadecimal): 0001**
**Description:** Detailed exception data was placed into the Activity Log.
**Minor Code:** CORBAMinorActivityLog
**Message ID:** CommonMessageActivityLogEntry

**Code (Hexadecimal): 0002 - 0006**
**Description:** Reserved

**Code (Hexadecimal): 0007**
**Description:** Unable to load message catalog %s, message %d.
**Minor Code:** CORBAMinorCommonCannotLoadCatalog
**Message ID:** CommonMessageCannotLoadCatalog

**Code (Hexadecimal): 0008**
**Description:** Unable to find message %d in message catalog %s.
**Minor Code:** CORBAMinorCommonCannotFindMessage
**Message ID:** CommonMessageCannotFindMessage

**Code (Hexadecimal): 0009**
**Description:** Severe Error, data was placed in the Error Log.
**Minor Code:** CORBAMinorCommonSevereError
**Message ID:** CommonMessageSevereError

**Code (Hexadecimal): 000A**
**Description:** Out of memory.
**Minor Code:** CORBAMinorCommonOutOfMemory
**Message ID:** CommonMessageOutOfMemory

**Code (Hexadecimal): 000B**
**Description:** Received Operation System Exception %s.
**Minor Code:** CORBAMinorCommonOSException
**Message ID:** CommonMessageOSException

**Code (Hexadecimal): 000C**
**Description:** Operating System call %s returned %d.
**Minor Code:** CORBAMinorCommonOSReturnCode
**Message ID:** CommonMessageOSReturnCode

**Code (Hexadecimal): 000D**
**Description:** Failure occurred when creating a thread.
**Minor Code:** CORBAMinorCommonThreadCreateFail
**Message ID:** CommonMessagethreadCreateFail

**Code (Hexadecimal): 000E**
**Description:** Failure timeout occurred when waiting on a thread
**Minor Code:** CORBAMinorCommonThreadWaitTimeout
**Message ID:** CommonMessageThreadWaitTimeout

**Code (Hexadecimal): 000F**
**Description:** Failure occurred when creating a semaphore.
**Minor Code:** CORBAMinorCommonSemaphoreCreateFail
**Message ID:** CommonMessageSemaphoreCreateFail

**Code (Hexadecimal): 0010**
**Description:** Failure occurred when requesting a semaphore.

**Minor Code:** CORBAMinorCommonSemaphoreRequestFail
**Message ID:** CommonMessageSemaphoreRequestFail

**Code (Hexadecimal): 0011**
**Description:** Failure occurred when releasing a semaphore.
**Minor Code:** CORBAMinorCommonSemaphoreReleaseFail
**Message ID:** CommonMessageSemaphoreReleaseFail

**Code (Hexadecimal): 0012**
**Description:** Failure occurred on semaphore signal.
**Minor Code:** CORBAMinorCommonSemaphorSignalFail
**Message ID:** CommonMessageSemaphoreSignalFail

**Code (Hexadecimal): 0013**
**Description:** Failure timeout occurred when waiting on a semaphore.
**Minor Code:** CORBAMinorCommonSemaphoreWaitTimeout
**Message ID:** CommonMessageSemaphoreWaitTimeout

**Code (Hexadecimal): 0014**
**Description:** Failure occurred when loading executable %s.
**Minor Code:** CORBAMinorCommonProgramLoadFail
**Message ID:** CommonMessageProgramLoadFail

**Code (Hexadecimal): 0015**
**Description:** Failure occurred when loading library %s.
**Minor Code:** CORBAMinorCommonDLLLoadFail
**Message ID:** CommonMessageDLLLoadFail

**Code (Hexadecimal): 0016**
**Description:** Failure occurred when creating a heap.
**Minor Code:** CORBAMinorCommonHeapCreateFail
**Message ID:** CommonMessageHeapCreateFail

**Code (Hexadecimal): 0017-0031**
**Description:** Reserved

**RELATED REFERENCES**
“Reading the Activity Log” on page 6

# Appendix H. APPC Messages

While it is running, Component Broker may issue messages related to APPC connections to tier-3 systems. To understand APPC messages, see the following messages descriptions. When searching a log for APPC-related messages, search for the string "ots_appc".

For some example APPC message scenarios, see “Example: APPC Messages in the Activity Log” on page 81.

**APPC support has been enabled in the transaction service.**
**Explanation:** The APPC support of the Transaction Service has been loaded in the server. This is DLL somtra1i.dll.
**System Action:** Server continues initializing.
**User Response:** This message is for information only. No action is required.

**The SNA library** *dllName***, version** *major***.***minor***, for** *productName* **has been loaded in the server.**
**Explanation:** The SNA library used for PAA APPC support has been loaded.
**System Action:** Server continues initializing.
**User Response:** Check that this is the SNA product that the server should be using.

**Unable to locate SNA library** *dllName***. Check that a supported SNA product has been installed on your machine and the path variables have been set up correctly. (For example, has the machine been rebooted since the SNA product was installed?) It is possible to specify the full path of the DLL using the environment variable** *variableName***.**
**Explanation:** The SNA library used for PAA APPC support cannot be loaded.
**System Action:** Server continues initializing. However, attempts to use PAA APPC result in CORBA::NO_IMPLEMENT exceptions being raised.
**User Response:** Verify the set up of the machine and, once corrected, restart the server.

**Unable to locate entry point** *name* **in SNA library** *dllName***. Check that a supported SNA product has been installed on your machine and the path variables have been set up correctly. (For example, has the machine been rebooted since the SNA product was installed?) It is possible to specify the full path of the DLL using the environment variable** *variableName***.**
**Explanation:** The SNA library used for PAA APPC support cannot be loaded.
**System Action:** Server continues initializing. However, attempts to use PAA APPC result in CORBA::NO_IMPLEMENT exceptions being raised.
**User Response:** Verify the set up of the machine and, once corrected, restart the server.

**APPC has been activated in this server. However, it is not supported on this operating system.**
**Explanation:** PAA APPC support is not available on this operating system.
**System Action:** Server continues initializing. However, attempts to use PAA APPC result in CORBA::NO_IMPLEMENT exceptions being raised.
**User Response:** Check later versions of Component Broker, because one of them may support PAA APPC on this operating system, and upgrade as necessary.

**The SNA library used for PAA APPC support returned ’***primaryRC***/***secondaryRC***’ from an APPC ’***verb***’ call.**

**Explanation:** The current APPC request *verb* is unsuccessful. '*primaryRC*/*secondaryRC*' are the hex values of the return codes from the call.
**System Action:** The current APPC request is unsuccessful. This may affect any associated transaction.
**User Response:** Look for other Component Broker messages in the activity log that occur at the same time as this, as they may indicate the cause of the problem. If no other messages exist, look up the symbolic names of the return codes in the sdk/win32/h/winrc.h header file located in the IBM Communication Servers install directory. Then look up the meaning of the return codes in the IBM Communication Programming manuals.

**Server** *serverName* **will send APPC requests from SNA local LU name '***luname***' which is defined with an alias name of '***alias***'.**
**Explanation:** This message shows the name that the server is using to register in the SNA network. This is called the local LU name. Remote systems should be configured to communicate with this local LU name. The alias is a nickname for the local LU name which is known only on this machine.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**Unable to complete transaction** *tranName* **(SNA LUWId** *luwId***) because partner SNA APPC system '***luname***' is currently unreachable. An attempt to complete the transaction will be made later.**
**Explanation:** This message warns that a local transaction is unable to complete because a partner SNA APPC system is unavailable.
**System Action:** The server continues to retry the connection to the partner SNA APPC system.
**User Response:** Attempt to restart the partner SNA APPC system.

**Format error detected in SNA presentation services (PS) header received from partner SNA APPC system '***luname***'. This partner SNA APPC system will no longer be included in the completion of transaction** *tranName* **(SNA LUWId** *luwId***).**
**Explanation:** This message reports that the local server and partner SNA APPC system are unable to communicate correctly to commit a transaction.
**System Action:** The server completes the transaction locally. Updates to data made in the partner SNA APPC system may be left locked and will have to be freed manually.
**User Response:** Turn on traces of SNA link and local server and retry the request. Contact IBM support.

**Format error detected in SNA presentation services (PS) header sent to partner SNA APPC system '***luname***'. This partner SNA APPC system will no longer be included in the completion of transaction** *tranName* **(SNA LUWId** *luwId***).**
**Explanation:** This message reports that the local server and partner SNA APPC system are unable to communicate correctly to commit a transaction.
**System Action:** The server completes the transaction locally. Updates to data made in the partner SNA APPC system may be left locked and will have to be freed manually.
**User Response:** Turn on traces of SNA link and local server and retry the request. Contact IBM support.

**Heuristic decision made locally for transaction** *tranName* **(SNA LUWId** *luwId***) will be passed to partner SNA APPC system** *luname* **and affect updates made by remote transaction program** *tpn* **on conversation** *correlator***.**

**Explanation:** This message reports part of the impact of a locally made heuristic decision.
**System Action:** The server completes the transaction locally and in the partner SNA APPC system.
**User Response:** Check for heuristic exceptions raised in other local servers involved in the transaction. Follow local procedures to correct data where these exceptions occur.

**Partner SNA APPC system '***luname***' made a heuristic decision to commit the updates made by transaction program '***tpn***' for transaction** *tranName* **(SNA LUWId** *luwId***). This decision was inconsistent with local transaction and heuristic damage has occurred.**
**Explanation:** This message reports the occurrence of heuristic damage caused by action taken in a partner SNA APPC system.
**System Action:** The server completes the transaction locally.
**User Response:** Follow local procedures to correct data as appropriate in the local servers and in the partner SNA APPC system.

**Partner SNA APPC system '***luname***' made a heuristic decision to rollback the updates made by transaction program '***tpn***' for transaction** *tranName* **(SNA LUWId** *luwId***). This decision was inconsistent with local transaction and heuristic damage has occurred.**
**Explanation:** This message reports the occurrence of heuristic damage caused by action taken in a partner SNA APPC system.
**System Action:** The server completes the transaction locally.
**User Response:** Follow local procedures to correct data as appropriate in the local servers and in the partner SNA APPC system.

**Partner SNA APPC system '***luname***' has reported that updates made by transaction program '***tpn***' for transaction** *tranName* **(SNA LUWId** *luwId***) have been partially committed and partially rolled back. Heuristic damage has occurred in the partner SNA APPC system.**
**Explanation:** This message reports the occurrence of heuristic damage caused by action taken in a partner SNA APPC system.
**System Action:** The server completes the transaction locally.
**User Response:** Follow local procedures to correct data as appropriate in the local servers and in the partner SNA APPC system.

**Data updated by remote transaction program** *tpn* **in SNA partner SNA APPC system** *luname* **under transaction** *tranName* **(SNA LUWId** *luwId***) is locked, waiting for an upstream server to end the transaction.**
**Explanation:** This message reports that a transaction in the local server is waiting for another server to send it the outcome of the transaction. This is taking longer than expected.
**System Action:** The server waits to complete the transaction.
**User Response:** Restart any servers that have been involved in the transaction and are currently unavailable.

**SNA APPC communication between local server** *name* **(local LU name** *luname***) and partner SNA APPC system** *luname* **has failed. This may be due to either the partner SNA APPC system or part of the network shutting down.**
**Explanation:** This message reports that either part of the SNA network, or a partner SNA APPC system is unavailable.
**System Action:** The server may wait to complete the transaction or may roll back

the transaction.
**User Response:** Restart the SNA network or partner SNA APPC system as required.

**Transaction** *tranName* **(SNA LUWId** *luwId***) will rollback because of a communication failure between the local server (local LU name** *luname***) and a partner SNA system** *luname***.**
**Explanation:** This message reports the reason why a transaction has rolled back.
**System Action:** The server may wait to complete the transaction or may roll back the transaction immediately.
**User Response:** Restart the SNA network or partner SNA APPC system as required.

**Partner SNA APPC system** *luname* **has rejected the request to start transaction program** *tpn* **with a sense code of** *code***.**
**Explanation:** The server has made an APPC request to a partner SNA APPC system which has been rejected. The reason for this is specified in the SNA architected sense code which appears in the message. This sense code was sent by the partner system in a Function Management Header (FMH) type 7.
**System Action:** The server continues. Further messages are logged that describe the consequences of the sense code.
**User Response:** Look for other messages that relate to this error and follow the instructions for them. Alternatively, look up the sense code in the SNA architecture to discover the cause of the failure.

**Unable to automatically define local LU name '***luname***' with an alias of '***alias***' in the SNA configuration. The most likely cause is that the alias is already associated with another LU name.**
**Explanation:** The SNA product is unable to add the configuration for the specified local LU name.
**System Action:** The server continues. However, subsequent APPC requests that use the local LU name will probably fail.
**User Response:** Start up the SNA Node configuration tool and define a local LU name profile with a different alias. Activate this configuration and then rerun the request.

**Unable to automatically define partner LU name '***luname***' with an alias of '***alias***' in the SNA configuration. The most likely cause is that the alias is already associated with another LU name.**
**Explanation:** The SNA product is unable to add the configuration for the specified partner LU name. The most likely cause is that the alias is already associated with another LU name.
**System Action:** The server continues. However, subsequent APPC requests that use the partner LU name will probably fail.
**User Response:** Start up the SNA Node configuration tool and define a partner LU name profile with a different alias. Activate this configuration and then rerun the request.

**Unable to define mode name '***mode Name***' in the SNA configuration.**
**Explanation:** The SNA product is unable to add the configuration for the specified mode name.
**System Action:** The server continues. However, subsequent APPC requests that use the mode name will probably fail.
**User Response:** Start up the SNA Node configuration tool and define a mode profile for this mode name. Activate this configuration and then rerun the request.

**Starting the SNA attach manager.**
**Explanation:** The attach manager feature of the SNA product has been started. This is needed by the server to receive inbound requests.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**Unexpected error** *number* **from the SNA attach manager call** *name***.**
**Explanation:** This message reports a bad return code from a SNA call.
**System Action:** The system continues. Further messages may be logged describing the consequences of the failure.
**User Response:** Look for other messages and follow the instructions for them.

**The SNA product** *name* **is running.**
**Explanation:** The SNA product used to send APPC requests to partner SNA systems is available.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**The SNA product** *name* **has stopped.**
**Explanation:** The SNA product used to send APPC requests to partner SNA systems is currently unavailable. APPC requests will fail until it is restarted.
**System Action:** The server continues.
**User Response:** Restart the SNA product.

**Unexpected error** *number* **from the SNA call** *name***.**
**Explanation:** This message reports a bad return code from a SNA call.
**System Action:** The system continues. Further messages may be logged describing the consequences of the failure.
**User Response:** Look for other messages and follow the instructions for them.

**A request has been received from partner SNA APPC system '***luname***' to run SNA transaction program '***tpn***'. This transaction program is not available on the local server** *name* **(local LU name '***luname***') and so the request has been rejected.**
**Explanation:** This message reports an unsuccessful inbound request.
**System Action:** The server continues. The request is rejected.
**User Response:** Check the configuration in the partner SNA APPC system as it is probably set up to send the request to the wrong LU name.

**Partner SNA APPC system '***luname***' is unable to process the 0x06F2 request required to negotiate transactions support. This may be a transient problem. Until it is resolved, transactions are unavailable between the partner SNA APPC system and local LU name '***luname***'.**
**Explanation:** This message indicates that the exchange log names (XLN) program (also known as CLS2 on CICS systems) is currently unavailable on the partner SNA APPC system. Transactions that involve the partner SNA APPC system will be unsuccessful until the XLN program is run successfully.
**System Action:** Component Broker is unable to commit transactions that involve communication with the partner SNA APPC system *LUname* until the 0x06F2 request is successful. The Component Broker server keeps retrying.
**User Response:** 1) Verify that the partner SNA APPC system supports sync level 2, and is correctly configured and running for sync level 2; 2) Verify that the SNA network is up; and 3) verify that the LU names that appear in the message are correct. If they are not correct, then you need to update the attributes of the Connection by using the System Manager user interface.

**Connection '*name*' contains unsupported values. Use the systems management EUI to correct them.**
**Explanation:** This message indicates errors in systems management configuration.
**System Action:** Requests that use the connection will fail.
**User Response:** Look for other messages that indicate which values are invalid and correct them before re-running the request.

**Unable to communicate over connection '*name*' because the local user is not authorized to use transaction program '*tpn*' on partner SNA APPC system '*luname*'.**
**Explanation:** An APPC request has been rejected by the partner SNA APPC system because the security information (userid and password) sent with the request is insufficient to be granted access.
**System Action:** The PAA APPC request fails. This does not affect the active transaction.
**User Response:** Change the security configuration either locally or in the partner SNA APPC system to allow the request to run.

**Unable to connect to partner SNA APPC system '*luname*' defined in connection '*name*' using local LU name '*luname*' and mode name '*modeName*'.**
**Explanation:** This message indicates a problem with the SNA configuration, the availability of the partner SNA APPC system, or the availability of part of the SNA network.
**System Action:** The PAA APPC request fails. This does not affect the transaction.
**User Response:** Ensure that the partner SNA APPC system, and the network to connect to it, are correctly configured and available, then retry the request.

**There has been a communications failure between partner SNA APPC system '*luname*' defined in connection '*name*' and local LU name '*luname*'.**
**Explanation:** This message reports that either part of the SNA network, or a partner SNA APPC system is unavailable.
**System Action:** The server may wait to complete the transaction or may roll back the transaction.
**User Response:** Restart the SNA network or partner SNA APPC system as required.

**Transactions are available between server *name* (local LU name '*luname*') and partner SNA APPC system '*luname*'.**
**Explanation:** This message indicates that the local server has successfully negotiated transactions support with the partner SNA APPC system.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**Transactions are not available between local server *name*(local LU name *luname*) and partner SNA APPC system '*luname*' because the partner SNA APPC system still has outstanding transactions to complete and the local server no longer has the same transaction log as it was using before.**
**Explanation:** This message indicates that the local server is unable to negotiate transaction support with the partner SNA APPC system.
**System Action:** The server continues. PAA APPC requests to the partner SNA APPC system will fail.
**User Response:** Take appropriate actions in the partner SNA APPC system to complete the transaction.

**Transactions are not available between local server *name* (local LU name *luname*) and partner SNA APPC system '*luname*' because the local server still**

**has outstanding transactions to complete and the partner SNA APPC system no longer has the same transaction log as it was using before.**
**Explanation:** This message indicates that the local server is unable to negotiate transaction support with the partner SNA APPC system.
**System Action:** The server continues. PAA APPC requests to the partner SNA APPC system will fail.
**User Response:** Shut down the local server. Change the server attributes to restrict the number of retries made by the Transaction Service and set the heuristic direction to the outcome required for the transactions. Then restart the server and wait for the transactions to complete.

**A 0x06F2 request to negotiate transactions support has been received from partner SNA APPC system '***luname***' which contains unexpected data or conversation attributes. Transactions are consequently unavailable between the partner SNA APPC system and local LU name '***luname***'.**
**Explanation:** This message indicates that part of a transaction that experienced a failure cannot be completed.
**System Action:** The server continues. The transaction remains incomplete.
**User Response:** Turn on traces of SNA link and local server and retry the request. Contact IBM support.

**Completed resynchronization of transaction** *name* **(SNA LUWId** *luwId***) for conversation '***correlator***'. The partner SNA APPC system '***luname***' was in '***name***' state and the local server (local LU name '***luname***') was in '***name***'. The outcome of the transaction was commit.**
**Explanation:** This message indicates that part of a transaction that experienced a failure has been successfully completed.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**Completed resynchronization of transaction** *name* **(SNA LUWId** *luwId***) for conversation '***correlator***'. The partner SNA APPC system '***luname***' was in '***name***' state and the local server (local LU name '***luname***') was in '***name***'. The outcome of the transaction was rollback.**
**Explanation:** This message indicates that part of a transaction that experienced a failure has been successfully completed.
**System Action:** The server continues.
**User Response:** This message is for information only. No action is required.

**Unable to complete resynchronization of transaction** *name* **(SNA LUWId** *luwId***) for conversation '***correlator***' because the partner SNA APPC system '***luname***' was in '***name***' state and the local server (local LU name '***luname***') was in '***name***'. These states are inconsistent.**
**Explanation:** This message indicates that part of a transaction that experienced a failure can not be completed.
**System Action:** The server continues. The transaction remains incomplete.
**User Response:** Follow procedures for the local and partner system to manually complete the transaction.

**Local logical unit (LU) '***luname***' is already in use by another server. APPC Connections that specify this local LU name will be unavailable in this server.**
**Explanation:** This message indicates that two servers are sharing the same local LU name. This is not supported and while this situation exists the Component Broker Transaction Service cannot guarantee recovery between those servers that use this local LU name and tier-3 systems.

**System Action:** Requests involving APPC connections that specify this local LU name will be unsuccessful.
**User Response:**

1. Locate all servers using this local LU name and shut them down.
2. Restart each one in turn and check that the transaction service successfully initializes before shutting it down again.
3. Delete the transaction service log files for each of these servers.
4. Correct the System Management configuration so that each server uses different APPC Connections which also specify different local LU names.
5. Activate this new configuration and restart the servers.

**Logical Unit (LU) name '*luname*' either contains characters other than A-Z, 0-9, @, $, #, or it begins with a number. This is not allowed by the SNA architecture.**
**Explanation:** Part of the configuration of an APPC Connection is invalid.
**System Action:** The APPC Connection is unusable.
**User Response:** Correct the value in the System Management configuration, reactivate this configuration and restart the server.

**Mode name '*modename*' either contains characters other than A-Z, 0-9, @, $, #, or it begins with a number. This is not allowed by the SNA architecture.**
**Explanation:** Part of the configuration of an APPC Connection is invalid.
**System Action:** Any APPC Connection specifying this value is unusable.
**User Response:** Correct the value in the System Management configuration, reactivate this configuration and restart the server.

**Mode name '*modename*' is reserved for SNA service requests and should not be used by PAA APPC applications, because this can cause deadlocks in the SNA network.**
**Explanation:** Part of the configuration of an APPC Connection is invalid.
**System Action:** None.
**User Response:** Change the value of the mode name in the APPC Connection in the System Management configuration. Reactivate this configuration and restart the server.

**Local LU name '*luname*' is defined in IBM Communication Server with the syncpoint support disabled. Please delete this local LU name from the Communication Server configuration and recycle (stop then start) the SNA node using the Communication Server Node Operations application. Component Broker will then define the local LU to Communication Server with the correct settings when the local LU name is next referenced by the server.**
**Explanation:** Part of the configuration of an APPC Connection is invalid.
**System Action:** Any APPC Connection specifying this value is unusable.
**User Response:** Remove the local LU name from the Communication Server configuration and restart the SNA Node.

**Component Broker is unable to use the local CP name as a local LU name because it cannot enable syncpoint support on the CP name. Please change the local LU name '*luname*' in the Component Broker APPC Connection definition so that it is different from the CP name configured in IBM Communication Server.**
**Explanation:** Part of the configuration of an APPC Connection is invalid.
**System Action:** Any APPC Connection specifying this value is unusable.

**User Response:** Change all APPC connections that specify this value of the local LU name so that they are using a different local LU name. Reactivate this configuration and restart the server.

**A request to allocate a session between local LU name** *luname* **and partner LU name** *luname* **using mode name '***modename***' has timed out after x seconds. This suggests that either there are insufficient sessions defined for the mode group or there is a problem in the partner SNA APPC system which is preventing the other PAA APPC requests from completing.**
**Explanation:** The SNA network is unable to process the current request due to limited resources.
**System Action:** The PAA APPC request fails which may cause the current transaction to rollback.
**User Response:** Check the state of the partner SNA APPC System and, if necessary, increase the number of sessions available on the mode group.

**A request to receive data from the partner SNA APPC system has timed out after x seconds. This suggests there is a problem in the partner SNA APPC system that is preventing this PAA APPC request from completing.**
**Explanation:** The SNA network is unable to process the current request due to limited resources.
**System Action:** The PAA APPC request fails which may cause the current transaction to rollback.
**User Response:** Check the state of the partner SNA APPC System and, if necessary, increase the number of sessions available on the mode group.

**The APPC Connection between local LU** *luname* **and partner LU** *luname* **was recovered from the Transaction Service log.**
**Explanation:** Information about a partner SNA APPC system which was connected to by this server on a previous occasion has been restored from the Transaction Service log.
**System Action:** Server continues initializing.
**User Response:** This message is for information only. No action is required.

**The APPC Connection between local LU** *luname* **and partner LU** *luname* **has been deleted from the Transaction Service log.**
**Explanation:** Information about a partner SNA APPC system which was connected to by this server on a previous occasion has been marked to be deleted from the Transaction Service log. The delete will actually happen during the next keypoint of the Transaction Service partner log.
**System Action:** Server continues.
**User Response:** This message is for information only. No action is required.

**Unable to recover outstanding transactions as the** *name* **component has not been enabled in this server.**
**Explanation:** A Component Broker feature which was enabled during a previous run of the server has now been disabled. However this feature is needed to initialize the transaction service.
**System Action:** Server continues to initialize. However new transactions can not be started.
**User Response:** Shut down the server, enable the missing feature and restart the server.

**Creating a new transaction service log** *logfilename* **for server** *serverName* **as the log used the last time the server was started is missing. This may cause data integrity problems if there were outstanding transactions when the**

**server terminated.**
**Explanation:** The transaction service is creating the files necessary to record details of any new transactions created or used in this server.
**System Action:** The transaction service continues with initialization.
**User Response:** Verify why the transaction log files are missing. If the log directory name has been changed and the log files have not been copied to the new directory, stop the server before new transactions are started, copy the log files to the new directory and restart the server. Use the activity log messages produced as the server starts to check that the log files have been opened successfully.

**The transaction service in server** *serverName* **is terminating.**
**Explanation:** The transaction service is ending in the server. This normally happens at server shutdown. However, if this message appears before the server is shutting down then an application has called CORBA::release() too often on the CosTransactions::Current object. In the latter case this message will probably be followed by a handleSignal() exception on the next request that uses transactions.
**System Action:** The transaction service shuts down and becomes unusable.
**User Response:** If the server is not shutting down when this message appears check all uses of the CosTransactions::Current object by your applications and correct the errors.

**Transaction** *name* **has rolled back.**
**Explanation:** An application requested a commit() operation for the transaction. However, the transaction service has rolled back the transaction due to an error detected by one of the components involved in the transaction.
**System Action:** The transaction service has already rolled back the transaction.
**User Response:** Check the activity log for other messages concerning this transaction.

**The current transaction is** *name***.**
**Explanation:** This message displays the transaction name. It is used to provide additional information when an error occurs and the application is using a transaction.
**System Action:** The server continues.
**User Response:** Check the activity log for other messages that occur on the same thread.

**The current transaction** *name* **is in** *state* **state.**
**Explanation:** This message displays the transaction name and state. It is used to provide additional information when an error occurs and the application is using a transaction.
**System Action:** The server continues.
**User Response:** Check the activity log for other messages that occur on the same thread.

**Transaction service log contains information about transaction** *name* **which was in** *transactionState* **state when the server terminated.**
**Explanation:** The server is starting up which involves reading the transaction service log to recover transactions that did not complete before the server shut down last time. Transaction *name* is one of these transactions. The *transactionState* value indicates which stage of processing had been reached and whether the transaction should be committed or rolled back by the transaction service.
**System Action:** The transaction service continues recovery.
**User Response:** None, this message is for information only.

**Recovered transaction *name* (SNA LUWId *luwId*) involved a connection to partner SNA APPC system '*partnerLUname*'.**
**Explanation:** The server is starting up which involves reading the transaction service log to recover transactions that did not complete before the server shut down last time. Transaction *name* is one of these transactions. Some of the work for transaction *name* occurred on the *partnerLUname* SNA system and this system may need to be contacted in order to complete the transaction.
**System Action:** The transaction service continues recovery.
**User Response:** Ensure SNA connectivity between the local server and the partner SNA APPC system.

**The resynchronization transaction program (0x06F2) is not recognized by partner SNA APPC system '*luName*'. This may be caused by a configuration problem in the partner SNA APPC system. Alternatively, the partner SNA APPC system is not able to support distributed transactions (syncpoints) using synclevel 2 over SNA.**
**Explanation:** When Component Broker contacts a partner SNA APPC system, it requests the partner's resynchronization transaction program called 0x06F2 (or CLS2 in a CICS system) is run to compare recovery information in each system's transaction service log. This process is not currently possible because partner *luName* does not recognize the 0x06F2 transaction. This usually means partner SNA APPC system has not been (or can not be) configured to support distributed transactions (syncpoints) using synclevel 2 over SNA.
**System Action:** PAA APPC requests to partner SNA APPC system *luName* will be unsuccessful until this problem is resolved.
**User Response:** Use the documentation supplied with the partner SNA APPC system to change its configuration to support distributed transactions over SNA. (The partner's documentation may refer to this as syncpoint support or synclevel 2 or synclevel syncpoint support.)

**The resynchronization transaction program (0x06F2) for partner SNA APPC system '*luName*' is currently unavailable. This may be caused by a configuration problem in the partner SNA APPC system. Alternatively, a process which controls SNA transactions on behalf of the partner SNA APPC system (such as an Encina PPC Gateway) is not running.**
**Explanation:** When Component Broker contacts a partner SNA APPC system, it requests the partner's resynchronization transaction program called 0x06F2 (or CLS2 in a CICS system) is run to compare recovery information in each system's transaction service log. This process is not currently possible because the partner *luName*'s 0x06F2 transaction is unavailable.
**System Action:** PAA APPC requests to partner SNA APPC system *luName* will be unsuccessful until this problem is resolved.
**User Response:** Use the documentation supplied with the partner SNA APPC system to enable the 0x06F2 transaction program.

**The transaction program '*tpn*' in partner SNA APPC system '*luName*' is currently unavailable. This may be caused by a configuration problem in the partner SNA APPC system. Alternatively, a process which controls SNA transactions on behalf of the partner SNA APPC system (such as an Encina PPC Gateway) is not running.**
**Explanation:** A PAA APPC request for transaction program *tpn* was unsuccessful because the partner SNA APPC system does not recognize the transaction program name *tpn* which means it can not identify which program to run.
**System Action:** PAA APPC requests for transaction program *tpn* to partner SNA APPC system *luName* will be unsuccessful until this problem is resolved.
**User Response:** Use the documentation supplied with the partner SNA APPC

system to enable the *tpn* transaction program, or update the transaction program name specified in the Component Broker APPC Connection.

**The APPC support is unable to send another request to the partner SNA APPC system '***luName***' for the current transaction** *name* **(SNA LUWId** *luwId***) because there is response data from the previous request that has not yet been processed. This is usually due to an incomplete output buffer defined in the PAA APPC Transaction Object (TO) navigation. This problem may result in the transaction rolling back.**
**Explanation:** A PAA APPC request for transaction program *tpn* was unsuccessful, because the data send by the transaction program as a response to the previous request has not been processed. This is usually due to an incomplete output buffer defined in the PAA APPC Transaction Object (TO) navigation. This problem may result in the transaction rolling back.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Correct the implementation of the Transaction Object (TO) and rerun the request.

**The APPC support is unable to send another request to the partner SNA APPC system '***luName***' because there is response data from the previous request that has not yet been processed.**
**Explanation:** A PAA APPC request for transaction program *tpn* was unsuccessful, because the data send by the transaction program as a response to the previous request has not been processed. This is usually due to an incomplete output buffer defined in the PAA APPC Transaction Object (TO) navigation. This problem may result in the transaction rolling back.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Correct the implementation of the Transaction Object (TO) and rerun the request.

**Transaction program** *tpn* **in partner SNA APPC system '***luName***' has rolled back transaction** *name* **(SNA LUWId** *luwId***).**
**Explanation:** The work carried out on partner SNA APPC system *luName* for transaction program *tpn* has been rolled back. This will result in all of the work on all servers for transaction *name* being rolled back.
**System Action:** The transaction is rolled back.
**User Response:** Check diagnostic messages on the partner SNA APPC system to determine why the transaction rolled back in that system. Take any corrective action on that system, if required.

**Transaction program** *tpn* **or partner SNA APPC system '***luName***' has abnormally terminated the conversation to the local server** *serverName* **(local LU name** *luName***).**
**Explanation:** The conversation between the local server and partner SNA APPC system *luName* has been shut down abnormally by actions in the partner system. This may be due to the transaction program explicitly terminating the conversation because it detected an error, or the partner SNA APPC system may have ended the conversation because of a time out, or because transaction program *tpn* terminated before ending the conversation. If the conversation was started at synclevel 2 then this will cause the transaction to rollback.
**System Action:** The conversation is cleaned up locally. The transaction may rollback later.
**User Response:** Check diagnostic messages on the partner SNA APPC system to determine why the conversation was abnormally terminated. Take any corrective action, if required.

**The APPC support is unable to start an APPC conversation between local server** *serverName* **(local LU name** *luName***) and partner SNA APPC system '***luName***'. This may be caused by a configuration error. Alternatively, part of the SNA network or the partner SNA APPC system may not be running.**
**Explanation:** This message means that Component Broker is unable to communicate with the partner SNA APPC system. This may be because of a configuration error (such as a misspelt LU name or mode name). Alternatively, part of the SNA network, or the partner SNA system itself may not be running.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Check the SNA configuration and network and correct the problem. This is best approached by checking the values used in the Component Broker APPC connection and IBM Communication Server. If these values are correct, check that they have been activated in the IBM Communication Server Node Operations Application. Then check that the partner SNA APPC system is running and is configured to accept requests from the local LU name. Finally, work out the path (machines and systems) through the SNA network that should be used to connect to the partner SNA APPC system. Make sure they are activated correctly. Look for error messages generated in these machines and systems.

**The APPC support is unable to send an APPC request between local server** *serverName* **(local LU name** *luName***) and partner SNA APPC system '***luName***' because the SNA sessions are not bound at synclevel 2. This could be because the local LU name has been defined explicitly to IBM Communication Server (in which case, delete the Local LU definition and restart the SNA node). Alternatively, the partner SNA APPC system has not been set up to support distributed transactions (syncpoints) over SNA.**
**Explanation:** This message means that Component Broker is unable to send transactional requests to the partner SNA APPC system because of a configuration error. Either the local LU name is defined in the IBM Communication Server configuration which has the effect of disabling the support for syncpoints or the partner SNA system is not set up to support syncpoints.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Check the SNA configuration using IBM Communication Server's Node Configuration Application. If the local LU name is in the configuration then delete it and restart the SNA node. If the local LU name is not defined to IBM Communication Server then correct the configuration in the partner SNA APPC system.

**The APPC support is unable to send an APPC request between local server** *serverName* **(local LU name** *luName***) and partner SNA APPC system '***luName***' because the local SNA node is not running. Use the IBM Communication Server Node Operations application to start the SNA node.**
**Explanation:** This message means that Component Broker is unable to send APPC requests to the partner SNA APPC system because IBM Communication Server is not running.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Use IBM Communication Server's Node Operations Application to start the SNA node. Then rerun the PAA APPC request.

**Transaction program '***tpn***' in partner SNA APPC system '***luName***' has rejected a PAA APPC request because it is expecting the conversation type to be BASIC and the local APPC support uses MAPPED conversations.**
**Explanation:** This message means there is an incompatibility between the APPC conversations used by Component Broker and those used by the partner SNA APPC system. Component Broker uses MAPPED conversations, however, the partner SNA APPC system is expecting a BASIC conversation. This may only be

due to a configuration error in the transaction program definition for *tpn* in the partner APPC system. Alternatively, transaction program *tpn* may need to be enhanced to accept mapped conversations.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Correct the configuration or code for transaction program *tpn* and rerun the PAA APPC request.

**Partner SNA APPC system '***luName***' has rejected a PAA APPC request because an exchange log names (XLN) conversation is required. The APPC support will attempt an exchange log names. When this is successful, retry the request.**
**Explanation:** This message means that a PAA APPC request was unsuccessful because it was sent in the small time window between Component Broker checking the status of the network, the network failing and being restarted and an exchange log names request used to validate the reconnection of the two systems. This condition should happen very rarely and only in the cases where the partner SNA APPC system or SNA network is restarted automatically.
**System Action:** The PAA APPC request is unsuccessful. Component Broker runs an exchange log names conversation with the partner SNA APPC system so that subsequent PAA APPC requests succeed.
**User Response:** Check for the message that indicates the exchange log names is successful and then retry the PAA APPC request.

**The APPC support is unable to send an APPC request between local server** *serverName* **(local LU name** *luName***) and partner SNA APPC system '***luName***' using mode name ' <modename>' because either the mode name or the partner LU name is not defined to IBM Communication Server or the partner SNA APPC system does not recognize the mode name.**
**Explanation:** This message means that a PAA APPC request was unsuccessful because the partner SNA APPC system can not be connected using the partner LU name and mode name supplied. This may be due to incorrect values being specified in the Component Broker APPC Connection or the operator may have deleted some of the configuration from IBM Communication Server Node Operations. Alternatively, the partner SNA APPC system may not be running, or it may be incorrectly configured.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Check the SNA configuration and network and correct the problem. This is best approached by checking the values used in the Component Broker APPC connection and IBM Communication Server. If these values are correct, check that they have been activated in the IBM Communication Server Node Operations Application. Then check that the partner SNA APPC system is running and is configured to accept requests from the local LU name. Finally, work out the path (machines and systems) through the SNA network that should be used to connect to the partner SNA APPC system. Make sure they are activated correctly. Look for error messages in these systems.

**The APPC support is unable to process the following SNA buffer because it contains an unexpected value in byte position** *index***. The buffer is** *x* **bytes long.**
**Explanation:** This message means that Component Broker is unable to interpret some data received from the SNA network. Other messages will indicate the resource affected.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Turn on Component Broker Transactions trace and IBM Communication Server's I-Frames trace and rerun the request. Save the activity log and these traces and contact your support center.

**The APPC support is unable to process the following SNA buffer because it does not contain all of the data expected. The buffer is *x* bytes long.**
**Explanation:** This message means that Component Broker is unable to interpret some data received from the SNA network. Other messages will indicate the resource affected.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Turn on Component Broker Transactions trace and IBM Communication Server's I-Frames trace and rerun the request. Save the activity log and these traces and contact your support center.

**The APPC support is unable to process the following SNA buffer because it contains unexpected values.**
**Explanation:** This message means that Component Broker is unable to interpret some data received from the SNA network. Other messages will indicate the resource affected.
**System Action:** The PAA APPC request is unsuccessful.
**User Response:** Turn on Component Broker Transactions trace and IBM Communication Server's I-Frames trace and rerun the request. Save the activity log and these traces and contact your support center.

**The APPC support has received a request to commit a transaction from transaction program '*tpn*' in partner SNA APPC system 'LUName'. This is not supported and will result in transaction *name* (SNA LUWId *luwId*) being rolled back.**
**Explanation:** This message means that a PAA APPC request was unsuccessful because the transaction program *tpn* attempted to initiate the commit (SYNCPOINT) of transaction *name* rather than waiting until the Component Broker application requested that the transaction committed. This is not supported and is a programming error in transaction program *tpn*.
**System Action:** The transaction is rolled back.
**User Response:** Correct transaction program *tpn* so that it does not initiate commit processing.

**Transaction program '*tpn*' in partner SNA APPC '*luName*' issued the *verbName* APPC verb (issued as *cpicVerbName* on the CPI-C programming interface or *cicsCallName* on the CICS programming interface). This is not supported and the APPC conversation has been abnormally terminated."**
**Explanation:** This message means that a PAA APPC request was unsuccessful because the transaction program *tpn* used an APPC verb that is not supported by Component Broker.
**System Action:** The conversation is abnormally terminated by Component Broker.
**User Response:** Correct transaction program *tpn* so that it does not use unsupported APPC verbs.

**The transaction service log in server *serverName* has created a new extent file** *extentFileName*
**Explanation:** The given extent file has been created by the transaction service log.
**System Action:** The server continues.
**User Response:** No action required, this message is for information only

**Transaction program '*tpn*' in partner SNA APPC '*luName*' has sent more than the *x* bytes of data requested by the local application. This is not supported and the APPC conversation has been abnormally terminated."**
**Explanation:** This message means that a PAA APPC request was unsuccessful because the transaction program *tpn* send more data than Component Broker is expecting.

**System Action:** The conversation is abnormally terminated by Component $ Broker.
**User Response:** Correct the navigation code that calls transaction program *tpn* so that it requests the correct length of data.

# Appendix I. Security Messages

While it is running, Component Broker may issue messages related to security. The security service records the following messages in the error, activity, or trace logs.

Many of these messages indicate an internal problem with the security service or the Component Broker run time. If you encounter any errors like that, report the problem to IBM.

However, for those messages that may be relevant to you or your programmers, descriptions are given following the table listing the messages. To understand those messages, see the message descriptions.

| Number | Message |
|---|---|
| 10114 | Method method_name of class '%2$s' failed. (page 283) |
| 10115 | DCE call '%1$s' failed. (page 283) |
| 10237 | DCE Login is required for this application... |
| 10238 | Enter Principal Name: |
| 10239 | Enter Realm (Cell) Name: |
| 10240 | Enter Password: |
| 10241 | Enter Realm (Cell) Name: " |
| 10242 | < Default Realm (Cell) Name: %s > |
| 10243 | SECURITY ERROR: DCE/GSS apis: %1$s reported an error code (%2$d) which indicates %3$s. (page 283) |
| 10244 | SECURITY ERROR: _narrow() failed on %1$s object. (page 283) |
| 10245 | SECURITY ERROR: _create() failed on %1$s object. (page 283) |
| 10246 | SECURITY ERROR: Failed to get an ORB Current object. (page 283) |
| 10247 | SECURITY ERROR: Failed to get an ORB object. (page 283) |
| 10248 | SECURITY ERROR: Failed to get imagename. (page 283) |
| 10249 | SECURITY ERROR: Failed to construct imagepath. |
| 10250 | SECURITY ERROR: Failed to open cursor on CDS for target %1$s. |
| 10251 | SECURITY ERROR: Failed to get hostname from CDS for target %1$s. |
| 10252 | SECURITY ERROR: Failed to get keytab file from CDS for target %1$s. |
| 10253 | SECURITY ERROR: Failed to get principal name from CDS for target %1$s. |
| 10254 | SECURITY ERROR: Failed to get QOP from CDS for target %1$s. |
| 10255 | SECURITY ERROR: Undefined hostname from CDS. |
| 10256 | SECURITY ERROR: Undefined keytab file from CDS. |
| 10257 | SECURITY ERROR: Undefined principal name from CDS. |
| 10258 | SECURITY ERROR: Undefined QOP from CDS. |
| 10259 | SECURITY ERROR: Undefined principal/password for data system from CDS. (page 284) |
| 10260 | SECURITY ERROR: Unsupported QOP: %d. (page 284) |
| 10261 | SECURITY ERROR: Unsupported Security Association Option: %d. |
| 10262 | SECURITY ERROR: Invalid Message Type. (page 284) |

| Number | Message |
|---|---|
| 10263 | SECURITY ERROR: Session entry that was created previously is not found. |
| 10264 | SECURITY ERROR: NULL value of session entry encountered. |
| 10265 | SECURITY ERROR: Security Tagged Component has no corresponding entry from the session table. |
| 10266 | SECURITY ERROR: Target Server's uuid is not found. (page 284) |
| 10267 | SECURITY ERROR: Failure during the invocation of non_existent(). (page 284) |
| 10268 | SECURITY ERROR: Found Security Tagged Component that contains no Tagged Component data. |
| 10269 | SECURITY ERROR: No Security Service Context found in the Request object. |
| 10270 | SECURITY ERROR: No or Insufficient Security Service Context received. |
| 10271 | SECURITY ERROR: Expected GIOP::Request message type but GIOP::Reply message type is received instead. |
| 10272 | SECURITY ERROR: No Security Context is received. |
| 10273 | SECURITY ERROR: Failed to verify the signature of security context when reclaiming message. (page 284) |
| 10274 | SECURITY ERROR: Vault::init_security_context failed to create security context. |
| 10275 | SECURITY ERROR: Mutual authentication handshake failure. (page 284) |
| 10276 | SECURITY ERROR: No principal is received. |
| 10277 | SECURITY ERROR: No token is returned from Vault::accept_security_context. (page 285) |
| 10278 | SECURITY ERROR: Default credential is not set properly. |
| 10279 | SECURITY ERROR: Cannot open catalog: %1$s. |
| 10280 | SECURITY ERROR: Cannot close catalog: %1$s. |
| 10281 | SECURITY ERROR: Cannot allocate memory from the heap for %1$s. (page 285) |
| 10282 | SECURITY ERROR: Unexpected internal error in Security. |
| 10283 | SECURITY ERROR: Undefined principal/hostname in Credentials object for target %1$s. |
| 10284 | SECURITY ERROR: Credentials object with principal %1$s does not represent target %2$s. |
| 10285 | SECURITY ERROR: Security is not enabled for target %1$s. |
| 10286 | SECURITY ERROR: Authentication is not enabled for target %1$s. |
| 10287 | SECURITY ERROR: Failed to insert a session entry. |
| 10288 | SECURITY WARNING: DCE configuration file \"dce_cf.db\" not found. |
| 10289 | SECURITY WARNING: Principal's DCE credential is going to expire. Attempt to renew. |
| 10290 | SECURITY WARNING: Server's DCE credential is going to expire. Attempt to renew. |
| 10291 | SECURITY WARNING: No established DCE login context available during authentication. |

| Number | Message |
|---|---|
| 10292 | SECURITY WARNING: Unable to certify the authenticity of principal %1$s. |
| 10293 | SECURITY WARNING: Invalid login context from DCE for principal %1$s. (page 285) |
| 10294 | SECURITY WARNING: Failed to get imagename. |
| 10295 | SECURITY WARNING: Failed to construct imagepath. |
| 10296 | SECURITY WARNING: Failed to open cursor on CDS for target %1$s. |
| 10297 | SECURITY WARNING: Failed to get hostname from CDS for target %1$s. |
| 10298 | SECURITY WARNING: Failed to get keytab file from CDS for target %1$s. |
| 10299 | SECURITY WARNING: Failed to get principal name from CDS for target %1$s. |
| 10300 | SECURITY WARNING: Failed to get QOP from CDS for target %1$s. |
| 10301 | SECURITY WARNING: Failed to get Login Source from CDS for target %1$s. |
| 10302 | SECURITY WARNING: Undefined hostname from CDS. (page 285) |
| 10303 | SECURITY WARNING: Undefined keytab file from CDS. |
| 10304 | SECURITY WARNING: Undefined principal name from CDS. (page 286) |
| 10305 | SECURITY WARNING: Undefined QOP from CDS. (page 286) |
| 10306 | SECURITY WARNING: Default credential is not set properly. |
| 10307 | SECURITY WARNING: Unknown server name. No Tagged component is created. |
| 10308 | SECURITY WARNING: <EstablishTrustInClient> is not set. |
| 10309 | SECURITY WARNING: Principal closed/cancelled Login Panel. |
| 10310 | SECURITY WARNING: Improper Component Broker login configuration. |
| 10311 | SECURITY WARNING: LoginHelper::request_login failed to return Credentials object. |
| 10312 | SECURITY WARNING: No Tagged Component profile is found in the proxy object of request message. |
| 10313 | SECURITY WARNING: No Security Tagged Component is found in Tagged Component profile. |
| 10314 | SECURITY WARNING: No Security Context has been received. |
| 10315 | SECURITY WARNING: Target Server's uuid is not found. |
| 10316 | SECURITY WARNING: init_security_context failed to create security context. |
| 10317 | SECURITY WARNING: _narrow() failed for %1$s. |
| 10318 | SECURITY WARNING: _create() failed for %1$s. |
| 10319 | SECURITY SSL ERROR: Key Ring file %1$s is not found. (page 286) |
| 10320 | SECURITY SSL ERROR: Key Ring file %1$s cannot be opened. |
| 10321 | SECURITY SSL ERROR: Bad format of Key Ring file %1$s. (page 286) |
| 10322 | SECURITY SSL ERROR: Bad password from Key Ring file %1$s. (page 287) |
| 10323 | SECURITY SSL ERROR: Invalid timeout value %1$d. |

| Number | Message |
|---|---|
| 10324 | SECURITY SSL ERROR: Expired certificate. |
| 10325 | SECURITY SSL ERROR: Invalid or No SSL Cipher is provided. |
| 10326 | SECURITY SSL ERROR: No SSL certificate is found in Key Ring file %1$s. (page 287) |
| 10327 | SECURITY SSL ERROR: Bad certificate. |
| 10328 | SECURITY SSL ERROR: Unsupported certificate type. (page 287) |
| 10329 | SECURITY SSL ERROR: Initialization failed. |
| 10330 | SECURITY SSL ERROR: Permission denied. |
| 10331 | SECURITY SSL ERROR: Unknown failure. |
| 10332 | SECURITY SSL ERROR: SSL type cannot be determined. |
| 10500 | SECURITY TRACE: Invoke non_existent() on server %1$s (page 287) |
| 10501 | SECURITY TRACE: Complete non_existent() to server %1$s. |
| 10502 | SECURITY TRACE: Receive a new security service context from client %1$s. (page 288) |
| 10503 | SECURITY TRACE:Call DCE gss_init_sec_context<br>for Target Server= %1$s,<br>in Vault::init_security_context. (page 288) |
| 10504 | SECURITY TRACE: Call gss_accept_sec_context in<br>Vault::accept_security_context :<br>Server Name = %1$s (page 288) |
| 10505 | SECURITY TRACE: Session Entry with<br>Target Server= %1$s,<br>Thread ID= %2$d<br>is being established in the Client Vault Session Table.<br>I(Thread ID= %3$d) am going to wait. (page 288) |
| 10506 | SECURITY TRACE:Wake up other threads that are waiting for the availability of session entry of the Target Server<br>Target Server=%1$s. (page 288) |
| 10507 | SECURITY TRACE: Server name is %1$s. |
| 10508 | SECURITY TRACE: Search Session Entry with<br>Target Server= %1$s,<br>Server UUID= %2$s,<br>Thread ID= %3$d<br>in the Client Vault Session Table. |
| 10509 | SECURITY TRACE: Security is enabled in System Management CDS data store. |
| 10510 | SECURITY TRACE: Security is disabled in System Management CDS data store. |
| 10511 | SECURITY TRACE: DCE Security Context(Credential) is valid. |
| 10512 | SECURITY TRACE: DCE Security Context(Credential) is invalid. |
| 10513 | SECURITY TRACE: Session entry is not initialized or Session entry state is UNSECURE. (page 288) |
| 10514 | SECURITY TRACE: Context Status of Session entry is unknown. (page 288) |
| 10515 | SECURITY TRACE: Authenticate server %1$s with keytab file %2$s. |
| 10516 | SECURITY TRACE: Succeed to authenticate security name = %1$s password/keytab file = %2$s. |

| Number | Message |
|---|---|
| 10517 | SECURITY TRACE: Succeed to refresh server. |
| 10518 | SECURITY TRACE: Out bound message to %1$s has valid DCE Credential. |
| 10519 | SECURITY TRACE: Out bound message to %1$s has invalid DCE Credential. |
| 10520 | SECURITY TRACE: Fail to refresh server. |
| 10521 | SECURITY TRACE: Security Server DLL is loaded. |
| 10522 | SECURITY TRACE: Security Client DLL is loaded. |
| 10523 | SECURITY TRACE: Fail to authenticate security name = %1$s password/keytab file = %2$s. |

**10114: Method** *method_name* **of class** *class_name* **failed.**
**Explanation:** Security code fails in a method invocation, such as authenticate() method of the PrincipleAuthenticator class.
**User Response:** None.

**10115: DCE call** *call_name* **failed.**
**Explanation:** Security code failed in DCE API calls. This error can be caused by DCE configuration errors or DCE runtime errors.
**System Action:** The DCE API call name is shown in the log entry.
**User Response:**Check that DCE is configured and running properly.

**10243: SECURITY ERROR: DCE/GSS apis:** *api_name* **reported an error code (***return_code***) which indicates message.**
**Explanation:** Security code failed in DCE API call. This error can be caused by DCE configuration errors or DCE run-time errors. DCE API name *api_name*, its return status *return_code*, and DCE return message message are shown in the log entry.
**User Response:** Check that DCE is configured and running properly.

**10244: SECURITY ERROR: _narrow() failed on %1$s object.**
**Explanation:** Security code fails to narrow a base class to its subclass. This is a Component Broker system error. The subclass name is shown in the log entry.
**User Response:** None.

**10245: SECURITY ERROR: _create() failed on %1$s object.**
**Explanation:** Security code fails to create an instance of the class. This error may be due to insufficient memory resource. The class name is shown in the log entry.
**User Response:** None.

**10246: SECURITY ERROR: Failed to get an ORB Current object**
**Explanation:** Security code fails to get the Security Current object from the ORB object. This is a Component Broker system error. Security initialization code might fail to register its Current object with the ORB object.
**User Response:** None.

**10247: SECURITY ERROR: Failed to get an ORB object.**
**Explanation:** Security code fails to get the ORB object from the CORBA::ORB_init() call. This is a Component Broker system error. Security initialization code fails to register its Current object with the ORB object.
**User Response:** None.

**10248: SECURITY ERROR: Failed to get imagename.**
**Explanation:** Security code fails to get the CDS image name of the current process. The CDS database might be corrupted.
**User Response:** Check the integrity of the CDS database.

**10259: SECURITY ERROR: Undefined principal/password for data system from CDS.**
**Explanation:** *data system principal* and *data system password* should be defined when the get_mapped_security_info() of the IExtendedSecurity::Credentials interface is called.
**User Response:** Check that the *data system principal* and *data system password* attributes were set for the data system.

**10260: SECURITY ERROR: Unsupported QOP: %d.**
**Explanation:** Unsupported security QOP options were received in the Security Interceptors.
**User Response:** None.

**10262: SECURITY ERROR: Invalid Message Type.**
**Explanation:** The message type is neither GIOP::Request nor GIOP::Reply. This is a Component Broker system error.
**User Response:** None.

**10263: SECURITY ERROR: Session entry that was created previously is not found.**
**Explanation:** A session entry that existed previously was lost from the Session Table. This is a Component Broker system error.
**User Response:** None.

**10265: SECURITY ERROR: Security Tagged Component has no corresponding entry from the session table.**
**Explanation:**A session entry is not found for a tag component when the client receives a response message in the interceptor. This is a Component Broker system error.
**User Response:** Not available.

**10266: SECURITY ERROR: Target Server's uuid is not found.**
**Explanation:** The target server UUID is not found in the target server proxy object. The target server is recognized by its UUID in the client process. This is a Component Broker system error.
**User Response:** None.

**10267: SECURITY ERROR: Failure during the invocation of non_existent().**
**Explanation:** The non_existent method invocation is used to establish security association between the client and the server. The failure of the non_existent method invocation means that the authentication process fails in the server.
**User Response:** Make sure that the client process is security enabled and the client has a valid DCE credential.

**10273: SECURITY ERROR: Failed to verify the signature of security context when reclaiming message.**
**Explanation:** A message exchanged between a client and server, or between two servers, could not be reclaimed. This could indicate that a rogue is tampering with the message traffic between these two points.
**User Response:** Check the message traffic to see if it is being tampered with.

**10275: SECURITY ERROR: Mutual authentication handshake failure.**
**Explanation:** The client principal and server principal could not be mutually authenticated. This may be the result of an internal processing error, or possibly a transient failure in the communication flow. However, it is more likely to indicate that either the client principal or the server principal have invalid authentication information, in which case this message should be accompanied by another message indicating a log-in failure. The client principal authentication errors can be caused by user errors. The server principal authentication errors are internal processing errors.
**User Response:** Check for any other messages indicating a log-in failure. If there are none, check that the authentication information for the client principal and server principal is valid.

**10277: SECURITY ERROR: No token is returned from Vault::accept_security_context.**
**Explanation:** Server fails to accept security context. It is caused by the failure in the DCE gss_accept_sec_context() call. The server fails to accept security context.
**User Response:** None.

**10281: SECURITY ERROR: Cannot allocate memory from the heap for <object name>.**
**Explanation:** This usually indicates that your host is running out of available memory. This may be due to having too many applications open at the same time, a memory leak in one of your applications, or simply that you don't have enough memory installed on your host.
**User Response:** Start by closing, and possibly restarting some of your applications. If necessary, install more memory on your host, or move some of your applications to another host.

**10293: SECURITY WARNING: Invalid login context from DCE for principal.**
**Explanation:** The DCE login context for the client principal is invalid. This is usually due to the context having expired. A Component Broker application server can become a DCE client when it invokes a method on the downstream server. A client's DCE login context will be automatically refreshed if its login source is in the DCE key table file. Otherwise, a panel will be prompted for user ID and password.
**System Action:** Component Broker will automatically attempt to refresh the credential, or solicit the userid and password from the client principal to form a new credential.
**User Response:** Most often, this warning can be ignored. However, if it recurs frequently there may be a problem with your DCE setup, or this may also be an indication that someone attempt to subvert the Component Broker security system.

**10302: SECURITY WARNING: Undefined hostname from CDS.**
**Explanation:** The host name defined in the Component Broker system management configuration is incorrect.
**User Response:** Check that Component Broker was installed properly onto the host. If needed, use the System Manager user interface to reconfigure your host with its correct host name or reinstall Component Broker onto the host.

**10303: SECURITY WARNING: Undefined keytab file from CDS.**
**Explanation:** The keytab file that you specified for the server in your system management Configuration does not exist. Because the keytab file is created automatically by Component Broker when the server is configured, this message usually means the keytab file has been subsequently deleted or renamed, or that the Configuration was changed to refer to the wrong keytab file. In this case, Component Broker will resort to using the default keytab file. However, if it is unable

to authenticate the server with the default keytab file, then this message should be accompanied by another message indicating that server authentication has failed.
**User Response:** Check for another message indicating that server authentication has failed. Try using the DCE administration tools to recreate the keytab file, and ensure the keytab file attribute defined on the Server model is set to refer to that keytab file.

**10304: SECURITY WARNING: Undefined principal name from CDS.**
**Explanation:** The security name specified for the server in your system management Configuration has not been set.
**User Response:** Use the System Manager user interface to set the security name attribute for the Server model then activate the Configuration again.

**10305: SECURITY WARNING: Undefined QOP from CDS.**
**Explanation:** This indicates that the system management configuration database (CDS) is corrupted. All **standard export QOP model** or **standard perform QOP model** options in the Security Service notebook of the Server Image or the Client Image should be supported by Component Broker.
**User Response:** Use the System Manager user interface to set and activate the QOP-related attributes for your Server or Client Style, then activate the Configuration again.

1**0319: SECURITY SSL ERROR: Key Ring file** *file/class name* **is not found.**
**Explanation:**The keyring file or class specified for the server or client in your system management Configuration does not exist. The keyring file may have been deleted or renamed, or the keyring file setting may have been incorrect to begin with.
For Java clients, the keyring file attribute should specify a Java class. Therefore the keyring file value in the Client Style model should not have any directory path or .class extension. Rather the keyring file attribute should be a normally formed Java class name, corresponding to the class name you assigned to the keyring class with KEYMAN or IKMGUI.
**User Response:** Locate the correct keyring file and either recreate it or rename it back to its correct name, or use the System Manager user interface to modify the Client Style or Server to set the keyring file attribute to the correct name, then activate the Configuration again.
For Java clients, check that the keyring class can be found in the classpath or, if you are using a Java Applet client, in the CODE, CODEBASE, or ARCHIVE attributes of the APPLET tag in the HTML document that initiated the applet.

**10320: SECURITY SSL ERROR: Key Ring file** *file/class name* **can not be opened.**
**Explanation:** The keyring file or class could not be opened. This is probably due to an access violation for the file; the identify under which the client or server is executing is not authorized to access the keyring file. Check the permissions on the keyring file and make sure the client or server is able to read it.
**User Response:** Locate the correct keyring file and either recreate it or rename it back to its correct name, or use the System Manager user interface to modify the Client Style or Server to set the keyring file attribute to the correct name, then activate the Configuration again.
For Java clients, check that the keyring class can be found in the classpath or, if you are using a Java Applet client, in the CODE, CODEBASE, or ARCHIVE attributes of the APPLET tag in the HTML document that initiated the applet.

**10321: SECURITY SSL ERROR: Bad format of Key Ring file** *file/class name***.**
**Explanation:** The format of the keyring file or class is incorrect. The file may have

become corrupted.
**User Response:** If needed, recreate the keyring.

**10322: SECURITY SSL ERROR: Bad password from Key Ring file.**
**Explanation:** The password used to open the keyring file is incorrect. Either the keyring file has been changed — specifically the password on the keyring has been modified — or the password set for the client or server in your system management Configuration is incorrect.
**User Response:** Use the System Manager user interface to update the keyring password attribute of the Client Style or Server model with the correct password value, then reactivate your Configuration.

**10323: SECURITY SSL ERROR: Invalid timeout value** *timeoutvalue***.**
**Explanation:** The timeout value set for the client or server in your system management Configuration is invalid.
**User Response:** Use the System Manager user interface to update the SSL V3 Session Timeout attribute of the Client Style or Server model with the correct timeout value, then activate your Configuration again.

**10324: SECURITY SSL ERROR: Expired certificate.**
**Explanation:** The server certificate has expired. Most certificates are good for one year. When they have expired, they must be renewed with your certificate authority.
**User Response:** Follow the procedures specified by your certificate authority to renew or replace your server certificate, and then update it into your server keyring file. If you don't change the trust-basis for your server, then you should not have to change any client keyrings. However, the certificates for the trust-basis will eventually expire as well and, when they do, you will have to use the new or renewed certificate to update all of your client keyrings that cite that trust-basis.

**10326: SECURITY SSL ERROR: No SSL certificate is found in Key Ring file** *file/class name***.**
**Explanation:** The certificate for the corresponding server was not found in the keyring class file. This can occur at the server if you did not identify the server's certificate as being the default certificate in the keyring. Or this can occur at the client if a certificate for the trust-basis in the server is not included in the client's keyring.
**User Response:** Check that you have identified the server's certificate as being the default certificate in the server keyring and the certificate is included in the client's keyring.

**10327: SECURITY SSL ERROR: Bad certificate.**
**Explanation:** The certificate for the corresponding server is bad; somehow it has been corrupted.
**User Response:** You will usually have to recreate the certificate in the keyring. Follow the steps outlined in the *System Administration Guide*.

**10328: SECURITY SSL ERROR: Unsupported certificate type.**
**Explanation:** The certificate is of the wrong type. Component Broker only supports X.509v3 certificates.
**User Response:** Use the IKMGUI or KEYMAN administration tools to recreate the certificate.

**10500: Invoke Non_Existent Method**
**Explanation:** Client invokes non_existent method on the target server object to establish the security association between the client and the server.
**User Response:** None.

**10501: Complete Non_Existent Method**
**Explanation:** Client received a response message of the non_existent method from the server.
**User Response:** Not available.

**10502: Receive a New Security Context Request**
**Explanation:** Server received a non_existent method from the client.
**User Response:** None.

**10503: Call gss_init_sec_context**
**Explanation:** Client calls DCE GSS API, gss_init_sec_context, to initialize the DCE security context.
**User Response:** None.

**10504: Call gss_accept_sec_context**
**Explanation:** Server calls DCE GSS API, gss_accept_sec_context, to accept client's DCE security context.
**User Response:** None.

**10505: Multiple Treaded Client Request**
**Explanation:** Multiple threaded clients are attempting to establish the security context with the same target server.
**User Response:** None.

**10506: Security Association is Established**
**Explanation:** The security association between the client and the server is established.
**System Action:** Wake up the threads that are waiting on the security association.
**User Response:** None.

**10513: Unexpected Session Entry State**
**Explanation:** The session entry is in unknown state. This is a Component Broker system error.
**User Response:** None.

**10514: Incomplete Session Entry**
**Explanation:** The session entry is missing context status. This is a Component Broker system error.
**User Response:** Contact your IBM representative.

**RELATED TASKS**
"Troubleshooting Security Problems" on page 74
"Chapter 13. Security Service Trace" on page 55
Create and Install Server Certificates (*System Administration Guide*)
Place Server and Client Keyrings in your Enterprise (*System Administration Guide*)
Administer Accounts for Client and Server Principals (*System Administration Guide*)
Enable Security within a Configuration (*System Administration Guide*)

# Appendix J. Session Service Messages

When the Component Broker is running, it may issue messages related to its session services. To understand session service messages, see the following message descriptions.

**The top-level coordinator was not accessible from a sub-level coordinator, during an attempt to force session <sessionName> to end immediately in server <serverName>**
**Explanation:** The top-level coordinator was not accessible from a sub-level coordinator, during an attempt to force the session to end immediately. This exception can be generated if both the top-level and sub-level processes attempt to timeout a session at the same time, and in such a condition does not indicate an error. If this occurs, this message will be located between a CORBA::INV_OBJREF exception and a message indicating that the session has been timed out.
**System Action:** The server continues.
**User Response:** Verify that this message is a result of both the top-level and sub-level processes attempting to timeout the session. If this is not the case refer to previous errors in the activity log to determine the cause of the problem.

**An attempt was made to end session <sessionName> specifying an end mode not available from the current process**
**Explanation:** A sub level process has called Current::endSession with an end mode other than EndModeResetForce. Sub-level processes are restricted to calling Current::endSession with EndModeResetForce, but top-level processes can specify any end mode.
**System Action:** The server continues.
**User Response:** Verify the logic of the application code. Ensure that the caller should not be running in the top-level process where the session was created.

**Attempt to resume a transaction during the resume of session <sessionName> failed**
**Explanation:** While performing a session resume operation on the given session, a transaction within the scope of the session failed to resume successfully.
**System Action:** The session context is disassociated from the current thread.
**User Response:** Verify the logic of the application code. If necessary, refer to previous errors in the activity log to determine the cause of the problem.

**Session <sessionName> has timed out after <n> seconds in server <serverName>**
**Explanation:** Your application has set a time limit for session <sessionName> and this time limit has been reached. This may indicate that the server is waiting for a resource that is unavailable, or a deadlock has occurred in your application.
**System Action:** The session service resets the session.
**User Response:** Check that all of the servers involved in the session are operating correctly. In addition, check the use of locks in your application for possible deadlock situations.

**The operation has detected that session <sessionName> has been forced to end immediately**
**Explanation:** The session has been forced to end immediately. This can be caused by a timeout occurring, or as a result of the session being ended with the EndModeResetForce end mode specified. The operation which detected this condition is unable to continue.
**System Action:** The server continues.

**User Response:** Verify the cause of this message, and ensure that this is consistent with the logic of the application code.

**Session <sessionName> could not be imported into server <serverName>, as a problem was occurred whilst communicating with another server involved in the session**
**Explanation:** A problem occurred during communication with another server involved in the session. As a result, the session could not be imported and was forced to reset immediately.
**System Action:** The session is forced to reset.
**User Response:** Refer to previous errors in the activity log to determine why the session could not be imported.

**One or more of the resources involved in session <sessionName> did not checkpoint successfully. The checkpoint operation is incomplete**
**Explanation:** The checkpoint operation was incomplete because one or more resources involved in the session did not checkpoint successfully. This can occur if there was a problem communicating with another server involved in the session or if a resource itself was unable to checkpoint and raised the NotProcessed exception. Incomplete updates have occurred within the session.
**System Action:** The server continues.
**User Response:** Refer to previous errors in the activity log to determine why the resource was unable to checkpoint. Restart any servers that are currently unavailable. If necessary follow local procedures to correct data as appropriate in the local server(s).

**One or more of the resources involved in session <sessionName> did not reset successfully. The reset operation is incomplete**
**Explanation:** The reset operation was incomplete because one or more resources involved in the session did not reset successfully. This can occur if there was a problem communicating with another server involved in the session or if a resource itself was unable to reset and raised the NotProcessed exception. Incomplete updates have occurred within the session.
**System Action:** The server continues.
**User Response:** Refer to previous errors in the activity log to determine why the resource was unable to reset. Restart any servers that are currently unavailable. If necessary follow local procedures to correct data as appropriate in the local servers.

**One or more of the resources involved in session <sessionName> did not end successfully. The end operation is incomplete**
**Explanation:** The end operation was incomplete because one or more resources involved in the session did not end successfully. This can occur if there was a problem communicating with another server involved in the session or if a resource itself was unable to end and raised the NotProcessed exception. Incomplete updates have occurred within the session.
**System Action:** The server continues.
**User Response:** Refer to previous errors in the activity log to determine why the resource was unable to end. Restart any servers that are currently unavailable. If necessary follow local procedures to correct data as appropriate in the local servers.

# Appendix K. Transaction Service Messages

While it is running, the Transaction Service writes out a number of messages to the server's transaction log and activity log. To understand Transaction Service messages, see the following message descriptions:

**Opening new transaction service log <logfilename> in server <serverName>.**
**Explanation:** The Transaction Service is creating the files necessary to record details of any transactions created or used in this server.
**System Action:** The Transaction Service continues with the initialization.
**User Response:** None.

**Opening transaction service log <logfilename> in server <serverName>.**
**Explanation:** The Transaction Service is opening the files necessary to record details of any transactions created or used in this server.
**System Action:** The Transaction Service continues with the initialization.
**User Response:** None.

**Recovering incomplete transactions from the transaction service log for the server.**
**Explanation:** During server initialization, the Transaction Service detected that there were transactions running when the server was last terminated. These must be completed before new transactions can be created or used.
**System Action:** The Transaction Service attempts to contact the other servers involved in each of these transactions to determine whether the transaction should be committed or rolled back. Once the decision has been made, the chosen action is taken to complete the transaction. This might take some time if it needs to contact servers that are not currently available.
**User Response:** Ensure all servers that are involved in transactions with this server are running.

**All recovered transactions are now complete in server <serverName>.**
**Explanation:** The Transaction Service is unavailable in this process because it is not able to create a log to record information about transactions that are created or accessed in this server.
**System Action:** The server continues but transactions cannot be used.
**User Response:** If transactions are not used in this server, no action is needed. If transactions are required, ensure a directory for the log is correctly configured on the server. This log directory must be on a local disk with sufficient space to hold a separate set of log files for each server. Once the log configuration is correct, restart your server.

**The transaction service started successfully in server <serverName>.**
**Explanation:** The Transaction Service has successfully initialized inside the server and is now ready for new transactions to be created.
**System Action:** The server continues.
**User Response:** None.

**Client applications must link with client library <clientLibraryName> rather than the server library <serverLibraryName>.**
**Explanation:** The Transaction Service has been called in a process that has not been correctly initialized. If the process reporting this message is a client application, this application has been linked to the server library rather than the client library. If the process is a server, it has not been correctly configured to request server initialization.

**System Action:** The client application terminates immediately.
**User Response:** Link the client application with the client library <clientLibraryName> and not the server library <serverLibraryName>.

**Transaction <name> has timed out after <x> seconds in server <serverName>.**
**Explanation:** Your application has set a time limit for transaction <name> and this time limit has been reached. This might indicate that the server is waiting for a resource that is unavailable, or a deadlock has occurred in your application.
**System Action:** The Transaction Service rolls back the transaction.
**User Response:** Check that all the servers involved in the transaction are operating correctly. Check the use of locks in your application for possible deadlock situations.

**The transaction service in server <serverName> has made a heuristic commit decision for transaction <name>.**
**Explanation:** The Transaction Service in server <serverName> is unable to contact all the objects registered with the transaction within the number of retries permitted by the commitRetry attribute of the server. The Transaction Service has therefore taken a heuristic decision based on the setting of the heuristicDirection attribute for the server.
**System Action:** The Transaction Service commits the transaction in the local server and in any other server where it is controlling resources.
**User Response:** Restart any servers that are currently unavailable. This completes the transaction for any objects registered with the transaction that are located in these servers. Look at the action that has been taken by each of these objects and correct any problems in your application's data caused by the heuristic decision.

**The transaction service in server <serverName> has made a heuristic rollback decision for transaction <name>.**
**Explanation:** The Transaction Service in server <serverName> is unable to contact all the objects registered with the transaction within the number of retries permitted by the commitRetry attribute of the server. The Transaction Service has therefore taken a heuristic decision based on the setting of the heuristicDirection attribute for the server.
**System Action:** The Transaction Service rolls back the transaction in the local server and in any other server where it is controlling resources.
**User Response:** Restart any servers that are currently unavailable. This completes the transaction for any objects registered with the transaction that are located in these servers. Look at the action that has been taken by each of these objects and correct any problems in your application's data caused by the heuristic decision.

**The transaction service in server <serverName> has assumed a rollback vote for a resource that raised a heuristic exception during the prepare for transaction <name>.**
**Explanation:** An object that inherits from CosTransactions::Resource and that has registered with a CosTransactions::Coordinator object for transaction <name> has raised either a CosTransactions::HeuristicHazard or CosTransactions::HeuristicMixed exception during a prepare method call. The prepare method call usually returns a CosTransactions::Vote value. The Transaction Service has assumed the vote returned by this object is CosTransactions::VoteRollback, based on the setting of heuristicDirection configured for the server.
**System Action:** The Transaction Service rolls back the transaction.
**User Response:** Correct any problems in your application's data caused by the heuristic decision made in the CosTransactions::Resource object.

**The transaction service in server <serverName> has assumed a commit vote for a resource that raised a heuristic exception during the prepare for transaction <name>.**
**Explanation:** An object that inherits from CosTransactions::Resource and that has registered with a CosTransactions::Coordinator object for transaction <name> has raised either a CosTransactions::HeuristicHazard or CosTransactions::HeuristicMixed exception during a prepare method call. The prepare method call usually returns a CosTransactions::Vote value. The Transaction Service has assumed the vote returned by this object is CosTransactions::VoteCommit, based on the setting of heuristicDirection configured for the server.
**System Action:** The Transaction Service uses this vote to complete the two-phase commit.
**User Response:** Correct any problems in your application's data caused by the heuristic decision made in the CosTransactions::Resource object.

**The client library <clientLibraryName> could not be loaded.**
**Explanation:** The system was unable to load the client library <clientLibraryName>. This is usually because the library is not in one of the directories listed in your PATH environment variable. Alternatively, the file permissions for the library might not allow your userid to access the library.
**System Action:** The client application terminates immediately.
**User Response:** Check the libraries located in the directories listed in your PATH environment variable. Alter your PATH environment variable so that the correct client library is loaded.

**The transaction service in server <serverName> could not write the log name to the CDS.**
**Explanation:** The Transaction Service was unable to write a new log name to the CDS.
**System Action:** The server continues but transactions cannot be used.
**User Response:** Examine the activity log to find out why the CDS can not be written to.

**No new transaction service log names are available for server <serverName>.**
**Explanation:** The Transaction Service is unable to open a new log because all available log names (somtr000 to somtrFFF) are in use.
**System Action:** The server continues but transactions cannot be used.
**User Response:** None.

**The transaction service in server <serverName> is unable to maintain a log as there is insufficient space in the log directory.**
**Explanation:** The Transaction Service tried to create a new log or tried to write to an existing log. There was insufficient space in the file system which holds the log directory to allow this action.
**System Action:** The server continues but transactions cannot be used.
**User Response:** Make more space available in the Log directory.

**The transaction service log in server <serverName> has absorbed its log cushion file**
**Explanation:** The log cushion file has been removed in order to make space available for a new extent file. The file system which holds the log directory is getting low on free space.
**System Action:** The server continues.
**User Response:** Make more space available in the Log directory.

**The transaction service log in server <serverName> has restored its log cushion file.**
**Explanation:** A previously removed log cushion file has been re-created. This occurs in response to an increase in the amount of free space available on the file system which holds the log directory.
**System Action:** The server continues.
**User Response:** None.

**The transaction service in server <serverName> could not open its log. Error was <error>.**
**Explanation:** The required log could not be opened. This can occur if the log directory does not exist, or if one of the files that make up the servers log is open on another process.
**System Action:** The server continues but transactions cannot be used.
**User Response:** Make sure that the configured log directory exists. If it does exist, check that none of the files that make up the servers log are open on another process.

**RELATED REFERENCES**
"Appendix M. XA Messages" on page 305.

# Appendix L. Workload Management Messages

While it is running, Component Broker may issue messages related to workload management of client requests across application servers in controlled server groups.

To understand workload management messages, see the following message descriptions:

**1100 "System Error. A distributed set method has detected an invalid precondition (%1$d)."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1101 "System Error. The distributed set ElementMonitor state (%1$d) is invalid for check()."**
**Explanation:** Not available.
**User Response:**Contact your IBM Representative.

**1102 "System Error. The distributed set ElementMonitor cannot change state from %1$d to %2$d."**
**Explanation:** Not available.
User Response:Contact your IBM Representative.

**1200 "The object request broker has been server group enabled."**
**Explanation:** The issuing process is now able to apply workload distribution logic to remote workload managed objects.
**User Response:** For information only. If you did not intend for workload management to be enabled, you can disable the ORB's enhanced workload management (WLM) extension, as described in the *System Administration Guide*.

**1300 "The bind policy dynamic link library %1$s could not be loaded."**
**Explanation:** The bind policy dynamic link library <libraryName> could not be loaded.
**User Response:** Check that the bind policy library was correctly included as an additional executable with the application and has been successfully installed.

**1301 "The bind policy factory function %1$s could not be located in dynamic link library %2$s."**
**Explanation:** The bind policy factory function <functionName> either does not exist or has not been correctly exported from the library.
**User Response:** See information on writing Bind Policies in the Advanced Programming Guide.

**1302 "The bind policy factory function %1$s in dynamic link library %2$s failed."**
**Explanation:** The bind policy factory function <functionName> in dynamic link library <libraryName> failed.
**User Response:** Check earlier messages in the log and, if needed, the source of your bind policy. You may have made a mistake coding your bind policy, or other errors in the log may tell you what is happening.

**1303 "The bind policy method %1$s::init failed."**
**Explanation:** The bind policy method <methodName>::init failed.
**User Response:** Check earlier messages in the log and, if needed, the source of

your bind policy. You may have made a mistake coding your bind policy, or other errors in the log may tell you what is happening.

**1304 "The bind policy method %1$s::rankServers failed."**
**Explanation:** The bind policy method <methodName>::rankServers failed.
User Response: Check earlier messages in the log and, if needed, the source of your bind policy. You may have made a mistake coding your bind policy, or other errors in the log may tell you what is happening.

**1305 "The bind policy method %1$s::notifyChoice failed."**
**Explanation:** The bind policy method <methodName>::notifyChoice failed.
**User Response:** Check earlier messages in the log and, if needed, the source of your bind policy. You may have made a mistake coding your bind policy, or other errors in the log may tell you what is happening.

**1306 "Server %1$s is unavailable and will temporarily be excluded from server selection."**
**Explanation:** The workload distribution manager has detected a previous failure in processing a request on server <serverName>. An attempt will be made to automatically resend the request to another server in the controlled group, where possible.
**User Response** You may have deliberately made the server unavailable. Otherwise, check that the server is running and enabled for workload management. Perhaps check later messages to ensure that the request was serviced by another server.

**1307 "No servers are defined in server group %1$s."**
**Explanation:** No application servers that are members of the controlled server group <groupName> have registered with the SGCP server. The current request cannot be dispatched. An exception is raised.
**User Response:** Check the following points:

- You have configured some servers in the server group, and activated the system management Configuration containing the Server Group model
- At least one of the servers is running
- There were not any other errors logged during startup of the application servers (members of the server group).

**1308 "No servers are available in server group %1$s."**
**Explanation:** All of the server members of the controlled server group <groupName> that are registered with the SGCP have been marked unavailable. (See also message 1306 above.) The current request cannot be dispatched. An exception is raised.
**User Response:** Check the following points:

- You have configured some servers in the server group, and activated the system management Configuration containing the Server Group model
- At least one of the servers is running
- There were no other errors logged during startup of the application servers (members of the server group).

**1402 "The name space binding for Server Group Control Point %1$s can not be resolved."**
**Explanation:** The name space binding for Server Group Control Point <serverName> cannot be resolved.

**User Response:** Previous errors might indicate why the entry for the SGCP could not be resolved by the Name Server. Also check for errors logged by the SGCP server during startup.

**1403 "The server list for server group %1$s could not be determined."**
**Explanation:** The server list for server group <groupName> could not be determined.
**User Response:** Contact your IBM Representative.

**1404 "System Error. A returned token does not match any token in the cache."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1405 "No connection to the Server Group Control Point for server group %1$s is available."**
**Explanation:** No connection to the Server Group Control Point for server group <groupName> is available.
**User Response:** Check that the SGCP server is running. Check for other errors.

**1406 "The policy list for policy group %1$s could not be determined."**
**Explanation:** The policy list for policy group <groupName> could not be determined.
**User Response:** Contact your IBM Representative.

**1500 "The server has been activated as a member of server group %1$s."**
**Explanation:** The server has been activated as a member of server group <groupName>.
**User Response:** For information only.

**1501 "A corrupted or truncated server groups service context was detected."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1505 "Unable to register a WLM object reference."**
**Explanation:** A WLM object reference for an object configured for workload management could not be registered with the ORB. The object will be registered with a non-WLM object reference.
**User Response:** Check for any related errors that may have been logged earlier.

**1700 "Registration with the Server Group Control Point for server group %1$s has failed."**
**Explanation:** Registration with the Server Group Control Point for server group <groupName> has failed.
**User Response:** Check for any related errors that may have been logged earlier.

**1800 "Server Group Control Point for server group %1$s is initialized."**
**Explanation:** The SGCP Server is ready to control workload management for the server group <groupName>.
**User Response:** For information only.

**1801 "System Error. Attribute ServerGroupId is not configured for the Server Group Control Point."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1802 "System Error. Impossible state transition."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1803 "One or more servers in server group %1$s may continue to use expired configuration data for policy group %2$s."**
**Explanation:** As a result of reactivation, or other system management processes, the configuration information has changed for policy group <groupName>. The SGCP server has been unable to synchronize this change across all the servers that were actively registered.
If any registered server is still active, despite the SGCP server's attempt to contact it having failed, then that server may continue to accept requests from clients that have been based on the old configuration information for the policy group.
**User Response:** Check for other errors that may have been logged earlier showing this problem.

**1804 "System Error. Policy group %1$s has invalid state %1$d."**
**Explanation:** Policy group <groupName> has invalid state.
**User Response:** Contact your IBM Representative.

**1805 "Bind policy %1$s has an invalid C++ Class association (%2$s)."**
**Explanation:** The configuration information for Bind Policy <policyName> contains an invalid reference to a C++ Class object.
**User Response:** Check that the bind policy has been correctly configured with a C++ Class object representing the name of the C++ class implementing the bind policy.

**1806 "Bind policy %1$s has no valid C++ Class configuration."**
**Explanation:** The configuration of bind policy <policyName> is incomplete.
**User Response:** Check that the bind policy has been correctly configured with a C++ Class object representing the name of the C++ class implementing the bind policy.

**1807 "Exception caught storing attribute %1$s for Bind Policy %2$s."**
**Explanation:** An exception was caught whilst storing the attribute <attributeName> for Bind Policy <policyName>.
**User Response:** Check for previously logged errors.

**1808 "Exception caught storing attribute %1$s for Policy Group %2$s."**
**Explanation:** An exception was caught whilst storing the attribute <attributeName> for Policy Group<groupName>.
**User Response:** Check for previously logged errors.

**1809 "Exception caught storing attribute %1$s for Server Group Control Point."**
**Explanation:** An exception was caught whilst storing the attribute <attributeName> for the Server Group Control Point.
**User Response:** Check for previously logged errors.

**1810 "Server Group Control Point needs to add a new Policy Group."**
**Explanation:** The SGCP has processed a dynamic configuration update introducing a new Policy Group.
**User Response**: Not available.

**1811 "Policy Group %1$s needs to add a new Bind Policy."**
**Explanation:** The SGCP has processed a dynamic configuration update introducing a new Bind Policy into Policy Group <groupName>.
**User Response**: Not available.

**1812 "System Error. Exception caught creating datastore key to SGCP Image."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1813 "Server Group Control Point initialization terminated for server group %1$s."**
**Explanation:** An object required for workload management services could not be found. This message could just indicate that the server in question is in the process of being deleted. However it could also mean that the CDS is corrupt in some way.
**User Response:** Check the state of the server to see if it is being deleted. If required, check that the DCE CDS is not corrupted.

**1814 "System Error. Exception caught accessing Configuration Manager."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1815 "Active server %1$s has been removed from the server group."**
**Explanation:** As a result of activating a system management Configuration, a member server <serverName> has been removed from the server group. The SGCP server believed that server to be active and running.
**User Response:** Check that you meant to remove the server from the server group. Otherwise, if needed, reconfigure the server as a member of the server group then activate the system management Configuration again.

**1816 "Exception caught removing server information for server %1$s."**
**Explanation:** An exception was caught whilst removing server information for the server <serverName>.
**User Response:** Look for previous errors.There may be some problem with the CDS.

**1817 "System Error. Exception caught processing information on member servers."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1818 "System Error. Exception caught processing server registration."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1819 "Server registration for server %1$s failed. Server is not a member of the controlled server group."**
**Explanation:** A server has attempted to register its availability with the SGCP server, but the SGCP server does not accept that that server is a member of the server group. This may be a "race" condition that can be resolved by reactivation. It might indicate a more severe inconsistency in the CDS.
**User Response:** In the system management Configuration try removing the Server (member of group) from the Server Group, then activate the Configuration again. Then add the Server (member of group) back into the Server Group, then activate the Configuration again. If problems persist then contact your IBM Representative.

**1820 "System Error. Exception caught processing server deregistration."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1821 "Server deregistration for server %1$s failed. Server is not a member of the controlled server group."**
**Explanation:** A server has attempted to de-register its availability with the SGCP server, but the SGCP server does not accept that that server is a member of the server group. This may be a "race" condition that can be resolved by reactivation. It might indicate a more severe inconsistency in the CDS.
**User Response:** In the system management Configuration try removing the Server (member of group) from the Server Group, then activate the Configuration again. Then add the Server (member of group) back into the Server Group, then activate the Configuration again. If problems persist then contact your IBM Representative.

**1822 "A control point entry for this server may not have been unbound from the Name Service."**
**Explanation:** Not available.
**User Response:** Check the following entry in the DCE CDS and delete any entry found:
/workgroup/resources/server_groups/<server_group_name>/control_points/<server_name>

**1823 "System Error. Exception caught storing state data."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1824 "Server Group Control Point is unable to contact the Server Group Gateway server %1$s."**
**Explanation:** Server Group Control Point is unable to contact the Server Group Gateway server <serverName>.
**User Response:** Check earlier in the log for messages that may indicate some problem with Name Server access, or that the SGGW server is not running.

**1825 "System Error. Server registration for server %1$s failed. Server Group Control Point initialization is incomplete."**
**Explanation:** Server registration for server <serverName> failed. Server Group Control Point initialization is incomplete.
**User Response:** Contact your IBM Representative.

**1826 "System Error. Server deregistration for server %1$s failed. Server Group Control Point initialization is incomplete."**
**Explanation:** Server de-registration for server <serverName> failed. Server Group Control Point initialization is incomplete.
**User Response:** Contact your IBM Representative.

**1827 "System Error. Server Group Control Point initialization is incomplete. Client access is denied."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**1828 "Server information for server %1$s already exists and will be overwritten."**
**Explanation:** The SGCP server has found that information for a supposedly "new" server already existed in the SGCP datastore. Perhaps a server of the same name used to exist but was deleted and is now being reinstated and the original deletion did not clean up properly.

**User Response:** If you are confident that you really are adding a new server then this should be safe. If not then this might indicate a more serious CDS corruption problem.

**2000 "Internal unrecoverable error in server groups gateway server."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**2003 "Server groups gateway failed to bind server group name '%1$s' into the naming service."**
**Explanation:**The server groups gateway failed to bind the server group name <groupName> into the naming service. The Server Group Control Point Server will not start unless it can find this reference in the naming service.
**User Response:** Check that the CDS is correctly set up.

**2005 "Failed to create local server groups gateway intercept object."**
**Explanation:** Not available.
**User Response:** Contact your IBM Representative.

**2006 "Unknown exception when binding to naming service. Server will terminate."**
**Explanation:** Not available.
**User Response:** Inspect the activity log for other errors.

**2009 "Failed to read the data needed on request inbound to the Server groups gateway."**
**Explanation:** The Server group gateway server received a corrupted request.
**User Response:** If no other errors appear in the log, contact your IBM Representative.

**2010 "Exception when invoking outbound request from the Server groups gateway."**
**Explanation:** This failure indicates that the request redirected through the gateway server could not be dispatched to an application server in the controlled server group.
**User Response:** Check the log for other errors.

**2011 "Exception trapped when preparing outbound request from the Server groups gateway."**
**Explanation:** Not available.
**User Response:** The log should contain details of what the error actually was.

**2012 "System exception trapped calling method '%1$s' from the Server groups gateway."**
**Explanation:** A system exception was trapped when calling method <methodName> from the Server groups gateway. The exception will be returned to the client as a failure on the method call.
**User Response:** Inspect the log for errors.

**2013 "Unknown exception trapped calling method '%1$s' from the Server groups gateway."**
**Explanation:** An unknown exception was trapped when calling the method '<methodName> from the Server groups gateway. The exception will be returned to the client as a failure on the method call.
**User Response:** Inspect the log for errors.

**2014 "Error resolving parameter of type==’%1$s’, mode==’%2$s’ on method call ’%3$s’."**
**Explanation:** The information read from the Interface Repository tells the gateway server the signature of the method being called. While trying to resolve this particular type of parameter an error occurred.
**User Response:** Verify that the Interface Repository has the correct information for the method.

**2015 "Error resolving incoming parameters on method ’%1$s’, object type ’%2$s’."**
**Explanation:** The information read from the Interface Repository tells the gateway server the signature of the method being called. While trying to resolve the signature this error occurred.
**User Response:** Verify that the Interface Repository has the correct information for the method.

**2016 "Error getting an inout parameter return value on method ’%1$s’, object type ’%2$s’."**
**Explanation:** While using the information from the Interface Repository to get the value of a transmitted parameter an error occurred.
**User Response:** Check that the Interface Repository is correct or inspect the log for other errors.

**2017 "Error copying return value from method ’%1$s’, object type ’%2$s’. Value cannot be returned to the client."**
**Explanation:** The data returned from the application server in the server group could not be passed back to the client application.
**User Response:** Check that the Interface Repository correctly defines the return parameter types, and that the return value is correct.

**2051 "Server groups gateway cannot access Interface Repository. Requests cannot be forwarded."**
**Explanation:** An error occurred accessing the Interface Repository.
**User Response:** Check that the IR is correctly available to the host on which the gateway server is running.

**2052 "Server groups gateway cannot retrieve interface definition from Interface Repository for object ’%1$s’."**
**Explanation:** No entry could be found in the Interface Repository for the object type making the method call.
**User Response:** Not available.

**2053 "Server groups gateway cannot retrieve operation definition from Interface Repository for operation ’%1$s’ on object ’%2$s’."**
**Explanation:** Although information about this object can be found in the Interface Repository, no information about this method name could be found.
**User Response:** Add the information to the Interface Repository and rerun the application.

**2054 "Server groups gateway cannot create outgoing request object."**
**Explanation:** Not available.
**User Response:** Check the log for other errors.

**2055 "Server groups gateway did not get a response on method call ’%1$s’."**
**Explanation:** No response was returned on this method call, although one was

expected.
**User Response:** Check the log for other errors.

# Appendix M. XA Messages

When the Component Broker is running, it may issue messages relating to XA Resource Managers. To understand XA messages, see the following message descriptions:

**Attempt in server <serverName> to load switchLoadFile <switchLoadFileName> for the XA Resource Manager database <databaseName> failed. Check that the file exists and is in the current search path.**
**Explanation:** The server <serverName> failed to load the switch load file configured for the XA Resource Manager database <databaseName>.
**System Action:** The database is not available for use with this server.
**User Response:** Check that the switch load file exists and is available in the current search path.

**Error in server <serverName>. The switchLoadFile <switchLoadFileName> for the XA Resource Manager database <databaseName> is invalid because it does not contain the required function <functionName>.**
**Explanation:** The server <serverName> successfully loaded the switch load file configured for the XA Resource Manager database <databaseName>, but it does not contain the function <functionName> that is required to enable the server to obtain a pointer to the xa_switch_t structure of the XA Resource Manager.
**System Action:** The database is not available for use with this server.
**User Response:** Add the required function to the switch load file implementation. A sample switch load file is provided with Component Broker.

**Server <serverName> has successfully opened a connection to the XA Resource Manager database <databaseName>.**
**Explanation:** The server <serverName> successfully opened a connection to the XA Resource Manager database <databaseName>. The database is available for use with this server. (Future successful attempts to open connections to this database will not result in this message being issued).
**System Action:** None.
**User Response:** None.

**Attempt in server <serverName> to open a connection to the XA Resource Manager database <databaseName> failed with return code <xaReturnCode>.**
**Explanation:** The server <serverName> was unable to open a connection to the XA Resource Manager database <databaseName>. The return code received from xa_open was <returnCode>. The XA Resource Manager database is not available for use with the server.
**System Action:** None
**User Response:** If you do not require the database <databaseName> to run with this server, you can ignore this message. Otherwise, take the following steps to correct this problem:

1. Ensure that the Resource Manager is started.
2. Check that the database <databaseName> is correctly configured in the server configuration. In particular, ensure that the openString and the switchLoadFile are correct.
3. Ensure that the database exists on the Resource Manager.
4. Ensure that the server has the correct authority to access the database.

5. Check that the same Resource Manager is not configured more than once. The combination of openString and switchLoadFile must be unique for each XA Resource Manager configured.
6. Many Resource Managers have a configurable limit on the number of concurrent connections to the database. Check that the server has not exceeded this limit. This might be possible if it had many configured databases on the Resource Manager or if there were many applications accessing this database concurrently.
7. Establish why the server is unable to contact the Resource Manager database. Diagnostics provided for the Resource Manager may be useful. For example, if the Resource Manager Type is DB2 for NT, you should analyze the DB2 log file available in %DB2PATH%\%DB2INSTANCE%\db2diag.log for further information.

**No attempt was made by the server <serverName> to recover transactions on the XA Resource Manager database <databaseName> because a previous attempt to open a connection to this database failed.**
**Explanation:** Because the server <serverName> has failed to open a connection to the database <databaseName>, it has not attempted to recover any outstanding transactional work on the database. Outstanding transactional work relating to this server may have associated locks on the database which will not be released until this problem is fixed.
**System Action:** The server <serverName> will attempt to recover transactions on the database <databaseName> when it has successfully opened a connection to the database.
**User Response:** If you do not require the database <databaseName> to run with this server, and you are satisfied that no locks are held on the database as a result of previous outstanding transactional work by this server, you can ignore this message. Otherwise, refer to any previous messages in the log to establish why an attempt to open a connection to the database failed. Correct the problem based on the information provided in the previous messages.

**Attempt by the server <serverName> to recover transactions on the XA Resource Manager database <databaseName> failed with return code <returnCode>.**
**Explanation:** The server <serverName> may have failed to to recover all outstanding transactional work on the database. The return code received from xa_recover was <returnCode>. Outstanding transactional work relating to this server may have associated locks on the database which will not be released until this problem is fixed.
**System Action:** The server <serverName> will attempt to recover transactions on the database <databaseName> when it successfully opens a connection to the database. The XA Resource Manager database is not available for use with the server.
**User Response:** If you do not require the database <databaseName> to run with this server, and you are satisfied that no locks are held on the database as a result of previous outstanding transactional work by this server, you can ignore this message. Otherwise, establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> detected an error when attempting to rollback the transaction <tranName> on the XA Resource Manager database <databaseName> (configured as <configurationType> <imageName>). The**

**return code was <returnCode>. This error occurred during the database recovery processing.**
**Explanation:** The server <serverName> has failed to rollback the transaction <tranName> during an attempt to recover outstanding transactional work on the database. The return code received from xa_rollback was <returnCode>. This transaction may hold locks on the database which will not be released until this problem is fixed. The XA Resource Manager database is not available for use with the server.
**System Action:** The server <serverName> will keep attempting to rollback transaction <tranName> on database <databaseName> each time it successfully opens a connection to the database.
**User Response:** Establish why the server is unable to rollback this transaction on the Resource Manager database and correct the problem. Diagnostics for the Resource Manager may be useful.

**Server <serverName> is ignoring the configured XA Resource Manager database <databaseName> defined because it is already defined.**
**Explanation:** Two Resource Managers have been configured with the same unique identity. Each configured XA Resource Manager must be uniquely identified to a server. The unique identity is the databaseName of a rdbConnection image.
**System Action:** The server continues with initialization.
**User Response:** Check that the two configured XA Resource Managers refer to the same database. If they do, this message can be ignored. However, if they do not, access to the ignored Resource Manager will be impossible until the two XA resource managers have been assigned unique identities and the server has been restarted.

**Server <serverName> received an error from xa_end(TMSUCCESS) during completion processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to successfully complete a unit of work on an XA Resource Manager for the specified transaction. The request was xa_end and the return code received was <returnCode>.
**System Action:** The CosTransactions::Resource representing the updates to the XA Resource Manager database <databaseName> on server <serverName> for the specified transaction will vote to rollback the transaction. This will result in the transaction being rolled back.
**User Response:** Establish why the Resource Manager was unable to successfully complete the request and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an error from xa_prepare during two phase commit processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to successfully complete an xa_prepare during two phase commit processing for the specified transaction. The return code received was <returnCode>.
**System Action:** The CosTransactions::Resource representing the updates to the XA Resource Manager database <databaseName> on server <serverName> for the specified transaction will vote to rollback the transaction. This will result in the transaction being rolled back.
**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an error from xa_rollback during rollback processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to complete an xa_rollback request during transaction rollback processing. The return code received was <returnCode>.
**System Action:** The server action will depend on the return code received:

- If the return code indicates that the unit of work has been rolled back, the transaction can rollback successfully.
- If the return code indicates a heuristic problem, the transaction will report an appropriate heuristic outcome.
- If the return code indicates a Resource Manager problem, the transaction will report an outcome of HeuristicHazard.

**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an error from xa_commit during one phase commit processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to complete an xa_commit request during transaction one-phase commit processing. The return code received was <returnCode>.
**System Action:** The server action will depend on the return code received:

- If the return code indicates that the unit of work has been rolled back, the transaction will report an outcome of RolledBack.
- If the return code indicates a heuristic problem, the transaction will report an appropriate heuristic outcome.
- If the return code indicates a Resource Manager problem, the transaction will report an outcome of HeuristicHazard.

**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an error from xa_commit during two phase commit processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to complete an xa_commit request during two-phase commit processing of a transaction. The return code received was <returnCode>.
**System Action:** The server action will depend on the return code received:

- If the return code indicates that the unit of work has been committed, the transaction can commit successfully.
- If the return code indicates a heuristic problem, the transaction will report an appropriate heuristic outcome.
- If the return code indicates a Resource Manager problem, the transaction will report an outcome of HeuristicHazard.

**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an error from xa_forget during completion processing for transaction <tranName> on the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to complete an xa_forget request during transaction completion processing. The return code received was <returnCode>.
**System Action:** As the transaction outcome has already been determined, the server will ignore the error.
**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> received an <axRequest> from the XA Resource Manager database <databaseName>. The transaction service was unable to process the request, and returned <returnCode>.**
**Explanation:** The server <serverName> has failed to process a request received from the Resource Manager during normal processing. The request received was <axRequest> and the return code returned was <returnCode>.
**System Action:** None.
**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**Server <serverName> attempted to re-use a connection to XA Resource Manager database <databaseName>, but the connection has non-transactional work outstanding.**
**Explanation:** An XA Connection has been used to access an XA Resource Manager outside the scope of a transaction, and has an old unit of work in progress. The Transaction Service has received an ax_reg call from the XA Resource Manager to indicate the start of the unit of work, but it has not received an ax_unreg call to indicate that the unit of work is complete. If the unfinished unit of work relating to this server has associated locks on the database, these will not be released until the unit of work is completed.
**System Action:** The server <serverName> will be unable to use the connection for transactional work. If the unfinished unit of work has locks on the database, access to the database may be limited.
**User Response:** Establish the creator of the incomplete unit of work and make the changes required to ensure that the unit of work is completed. Note that if a request to a server causes a non transactional unit of work to be started (by accessing an XA database outside the scope of a transaction), it must complete that unit of work before returning. If the unfinished unit of work causes problems when accessing the XA Resource Manager <databaseName>, external facilities made available by the XA Resource Manager must be used to complete the unit of work.

**Server <serverName> received an error from xa_close when closing a connection to the XA Resource Manager database <databaseName>. The return code was <returnCode>.**
**Explanation:** The server <serverName> has failed to close a connection to an XA Resource Manager. The return code received was <returnCode>.

**System Action:** If possible, appropriate action will be taken to force closure of the connection. If this is not possible, the connection will be left and will not be re-used.
**User Response:** Establish why the Resource Manager was unable to successfully complete the request, and take appropriate action if required. Other messages in the log and diagnostics for the Resource Manager may be useful in identifying the problem.

**XA support has been enabled in the transaction service.**
**Explanation:** The XA support of the Transaction Service has been loaded in the server. This is DLL somtrx1i.dll.
**System Action:** Server continues initializing.
**User Response:** This message is for information only. No action is required.

**Heuristic decision made locally for transaction <tranName> (XID <xid>) will affect updates made to XA Resource Manager database <databaseName>.**
**Explanation:** This message reports part of the impact of a locally made heuristic decision.
**System Action:** The server completes the transaction locally and in the database server.
**User Response:** Check for heuristic exceptions raised in other local servers involved in the transaction. Follow local procedures to correct data where these exceptions occur.

**Data updated in XA Resource Manager database <database> under transaction <tranName> (XID <xid>) is locked, waiting for an upstream server to end the transaction.**
**Explanation:** This message reports that a transaction in the local server is waiting for another server to send it the outcome of the transaction. This is taking longer than expected.
**System Action:** The server waits to complete the transaction.
**User Response:** Restart any servers that have been involved in the transaction and are currently unavailable.

**An attempt to prepare transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName> failed. The transaction will be rolled back.**
**Explanation:** Transaction <tranName> is being committed but a problem has been encountered preparing data that was part of the transaction for commit. The data resides on database <databaseName>. As a result the transaction will be rolled back. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction is rolled back.
**User Response:** Check in the activity log for earlier messages that may indicated the cause of the problem and take action as appropriate for these messages.

**An attempt to prepare transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName> failed because an attempt to obtain a connection to the database failed. The transaction will be rolled back.**
**Explanation:** Transaction <tranName> is being committed, but a connection could not be obtained to database <databaseName> in order to prepare data that was part of the transaction for commit. As a result the transaction will be rolled back. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages

from the same transaction. <xid> is the name of the transaction that was provided to the database.<databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction is rolled back.
**User Response:** Check in the activity log for earlier messages that may indicated the cause of the problem and take action as appropriate for these messages.

**An attempt to commit transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName> failed because an attempt to obtain a connection to the database failed. This is reported to the transaction service as a transient error.**
**Explanation:** Transaction <tranName> is being committed, but a connection to database <databaseName> could not be obtained in order to commit the data. This is reported to the transaction service as a transient error. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction service will retry the commit according to the configured settings of the transaction service parameters "commit retry limit" and "retry restricted" for server <serverName>.
**User Response:** Check in the activity log for earlier messages that may indicate the cause of the problem, and take action as appropriate for these messages. If the problem persists such that the transaction service completes the configured number of retries without success, the updates to the database <databaseName> will remain uncommitted. In this situation, steps must be taken to commit the transaction using the facilities provided by the database provider. If this is not done, locks may remain on the database, and the database will retain information about the transaction in its log, which may result in problems due to the log being full.

**An attempt to rollback transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName> failed because an attempt to obtain a connection to the database failed. This is reported to the transaction service as a transient error.**
**Explanation:** Transaction <tranName> is being rolled back, but a connection to database <databaseName> could not be obtained in order to commit the data. This is reported to the transaction service as a transient error. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction service will retry the rollback according to the configured settings of the transaction service parameters "commit retry limit" and "retry restricted" for server <serverName>.
**User Response:** Check in the activity log for earlier messages that may indicate the cause of the problem, and take action as appropriate for these messages. If the problem persists such that the transaction service completes the configured number of retries without success, the updates to the database <databaseName> will remain active. In this situation, steps must be taken to rollback the transaction using the facilities provided by the database provider. If this is not done, locks may remain on the database, and the database will retain information about the transaction in its log, which may result in problems due to the log being full.

**An attempt to forget transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName> failed because an attempt to obtain a connection to the database failed. This is reported to the transaction service as a transient error.**
**Explanation:** Database <databaseName> reported a heuristic error during the completion of transaction <tranName>, and so must be directed to forget the transaction. However, a connection to the database could not be obtained in order to make the request to forget. This is reported to the transaction service as a transient error. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction service will retry the forget according to the configured settings of the transaction service parameters "commit retry limit" and "retry restricted" for server <serverName>.
**User Response:** Check in the activity log for earlier messages that may indicate the cause of the problem, and take action as appropriate for these messages. If the problem persists such that the transaction service completes the configured number of retries without success, the database must be told to forget the transaction using the facilities provided by the database provider. If this is not done, the database will retain information about the transaction in its log which may result in problems due to the log being full.

**A request has been received to prepare/commit transaction <tranName> (XID <xid>) on database <databaseName> in server <serverName>. However, due to previously reported failures the database has already been rolled back. As a result the transaction will be rolled back.**
**Explanation:** Transaction <tranName> is being committed, and a request has been made for updates to database <databaseName> to be committed. However, due to earlier reported problems, the database has already been rolled back. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The transaction will be rolled back.
**User Response:** Check in the activity log for earlier messages that may indicate why the database was previously rolled back, and take action as appropriate for these messages.

**A request to close a connection to database <databaseName> on server <serverName> was received when the connection was active for transaction <tranName> (XID <xid>). The database will be rolled back immediately and the resource representing the transaction will vote rollback when the transaction is completed.**
**Explanation:** The server has encountered problems using a connection to database <databaseName>, and has decided to close the connection. However, the connection is currently associated with transaction <tranName>, which means that the connection is currently being used to update the database. As a result, the database will be rolled back. <tranName> is the name of the transaction used internally by the transaction service. It is included so that this message can be associated with other messages from the same transaction. <xid> is the name of the transaction that was provided to the database. <databaseName> is the databaseName attribute of the appropriate RdbConnection image.
**System Action:** The database is rolled back immediately, but the transaction is

effectively marked 'rollback only'. As a result, any subsequent attempt to commit the transaction will result in a rollback.
**User Response:** Check in the activity log for earlier messages that may indicate why the decision was made to close the connection, and take action as appropriate for these messages.

**RELATED REFERENCES**
"Appendix K. Transaction Service Messages" on page 291

Part Number: SC09–2799–00

Printed in the United States of America

SC09–2799–00