TIBCO® Object Service Broker for z/OS

Installing and Operating

Software Release 6.0
July 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, The Power of Now, TIBCO Object Service Broker, and and TIBCO Service Gateway are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

The TIBCO Object Service Broker technologies described herein are protected under the following patent numbers:

Australia:	-	-	671137	671138	673682	646408
Canada:	2284250	-	-	2284245	2284248	2066724
Europe:	-	-	0588446	0588445	0588447	0489861
Japan:	-	-	-	-	-	2-513420
USA:	5584026	5586329	5586330	5594899	5596752	5682535

Contents

Preface

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme file for the availability of this software version on a specific operating system platform.

TIBCO® Object Service Broker is an application development environment and integration broker that bridges legacy and non-legacy applications and data.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Object Service Broker Documentation

The following documents form the TIBCO Object Service Broker documentation set:

Fundamental Information

The following manuals provide fundamental information about TIBCO Object Service Broker:

- *TIBCO Object Service Broker Getting Started* Provides the basic concepts and principles of TIBCO Object Service Broker and introduces its components and capabilities. It also describes how to use the default developer's workbench and includes a basic tutorial of how to build an application using the product. A product glossary is also included in the manual.
- *TIBCO Object Service Broker Messages with Identifiers* Provides a listing of the TIBCO Object Service Broker messages that are issued with alphanumeric identifiers. The description of each message includes the source and explanation of the message and recommended action to take.
- *TIBCO Object Service Broker Messages without Identifiers* Provides a listing of the TIBCO Object Service Broker messages that are issued without a message identifier. These messages use the percent symbol (%) or the number symbol (#) to represent such variable information as a rules name or the number of occurrences in a table. The description of each message includes the source and explanation of the message and recommended action to take.
- *TIBCO Object Service Broker Quick Reference* Presents summary information for use in the TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Shareable Tools* Lists and describes the TIBCO Object Service Broker shareable tools. Shareable tools are programs supplied with TIBCO Object Service Broker that facilitate rules language programming and application development.
- *TIBCO Object Service Broker Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Application Development and Management

The following manuals provide information about application development and management:

- *TIBCO Object Service Broker Application Administration* Provides information required to administer the TIBCO Object Service Broker application development environment. It describes how to use the administrator's workbench, set up the development environment, and optimize access to the database. It also describes how to manage the Pagestore, which is the native TIBCO Object Service Broker data store.
- *TIBCO Object Service Broker Managing Data* Describes how to define, manipulate, and manage data required for a TIBCO Object Service Broker application.
- *TIBCO Object Service Broker Managing External Data* Describes the TIBCO Object Service Broker interface to external files (not data in external databases) and describes how to define TIBCO Object Service Broker tables based on these files and how to access their data.
- *TIBCO Object Service Broker National Language Support* Provides information about implementing the National Language Support in a TIBCO Object Service Broker environment.
- *TIBCO Object Service Broker Object Integration Gateway* Provides information about installing and using the Object Integration Gateway which is the interface for TIBCO Object Service Broker to XML, J2EE, .NET and COM.
- *TIBCO Object Service Broker for Open Systems External Environments* Provides information on interfacing TIBCO Object Service Broker with the Windows and Solaris environments. It includes how to use SDK (C/C++) and SDK (Java) to access TIBCO Object Service Broker data, how to interface to TIBCO Enterprise Messaging Service (EMS), how to use the TIBCO Service Gateway for WMQ, how to use the Adapter for JDBC-ODBC, and how to access programs written in external programming languages from within TIBCO Object Service Broker.
- *TIBCO Object Service Broker for z/OS External Environments* Provides information on interfacing TIBCO Object Service Broker to various external environments within a TIBCO Object Service Broker z/OS environment. It also includes information on how to access TIBCO Object Service Broker from different terminal managers, how to write programs in external programming languages to access TIBCO Object Service Broker data, how to interface to TIBCO Enterprise Messaging Service (EMS), how to use the TIBCO Service Gateway for WMQ, and how to access programs written in external programming languages from within TIBCO Object Service Broker.

- *TIBCO Object Service Broker Parameters* Lists the TIBCO Object Service Broker Execution Environment and Data Object Broker parameters and describes their usage.
- *TIBCO Object Service Broker Programming in Rules* Explains how to use the TIBCO Object Service Broker rules language to create and modify application code. The rules language is the programming language used to access the TIBCO Object Service Broker database and create applications. The manual also explains how to edit, execute, and debug rules.
- *TIBCO Object Service Broker Managing Deployment* Describes how to submit, maintain, and manage promotion requests in the TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Defining Reports* Explains how to create both simple and complex reports using the reporting tools provided with TIBCO Object Service Broker. It explains how to create reports with simple features using the Report Generator and how to create reports with more complex features using the Report Definer.
- *TIBCO Object Service Broker Managing Security* Describes how to set up, use, and administer the security required for an TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Defining Screens and Menus* Provides the basic information to define screens, screen tables, and menus using TIBCO Object Service Broker facilities.
- *TIBCO Service Gateway for Files SDK* Describes how to use the SDK provided with the TIBCO Service Gateway for Files to create applications to access Adabas, CA Datacom, and VSAM LDS data.

System Administration on the z/OS Platform

The following manuals describe system administration on the z/OS platform:

- *TIBCO Object Service Broker for z/OS Installing and Operating* Describes how to install, migrate, update, maintain, and operate TIBCO Object Service Broker in a z/OS environment. It also describes the Execution Environment and Data Object Broker parameters used by TIBCO Object Service Broker.
- *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* Explains the backup and recovery features of OSB for z/OS. It describes the key components of TIBCO Object Service Broker systems and describes how you can back up your data and recover from errors. You can use this information, along with assistance from TIBCO Support, to develop the best customized solution for your unique backup and recovery requirements.

- *TIBCO Object Service Broker for z/OS Monitoring Performance* Explains how to obtain and analyze performance statistics using TIBCO Object Service Broker tools and SMF records
- *TIBCO Object Service Broker for z/OS Utilities* Contains an alphabetically ordered listing of TIBCO Object Service Broker utilities for z/OS systems. These are TIBCO Object Service Broker administrator utilities that are typically run with JCL.

System Administration on Open Systems

The following manuals describe system administration on open systems such as Windows or UNIX:

- *TIBCO Object Service Broker for Open Systems Installing and Operating* Describes how to install, migrate, update, maintain, and operate TIBCO Object Service Broker in Windows and Solaris environments.
- *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* Explains the backup and recovery features of TIBCO Object Service Broker for Open Systems. It describes the key components of a TIBCO Object Service Broker system and describes how to back up your data and recover from errors. Use this information to develop a customized solution for your unique backup and recovery requirements.
- *TIBCO Object Service Broker for Open Systems Utilities* Contains an alphabetically ordered listing of TIBCO Object Service Broker utilities for Windows and Solaris systems. These TIBCO Object Service Broker administrator utilities are typically executed from the command line.

External Database Gateways

The following manuals describe external database gateways:

- *TIBCO Service Gateway for DB2 Installing and Operating* Describes the TIBCO Object Service Broker interface to DB2 data. Using this interface, you can access external DB2 data and define TIBCO Object Service Broker tables based on this data.
- *TIBCO Service Gateway for IDMS/DB Installing and Operating* Describes the TIBCO Object Service Broker interface to CA-IDMS data. Using this interface, you can access external CA-IDMS data and define TIBCO Object Service Broker tables based on this data.
- *TIBCO Service Gateway for IMS/DB Installing and Operating* Describes the TIBCO Object Service Broker interface to IMS/DB and DB2 data. Using this interface, you can access external IMS data and define TIBCO Object Service Broker tables based on it.

- *TIBCO Service Gateway for ODBC and for Oracle Installing and Operating*
  Describes the TIBCO Object Service Broker ODBC Gateway and the TIBCO Object Service Broker Oracle Gateway interfaces to external DBMS data. Using this interface, you can access external DBMS data and define TIBCO Object Service Broker tables based on this data.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	`MyCommand {param1 param2} | {param3 param4}`
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	`MyCommand {param1 | param2} {param3 | param4}`
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts, a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access All TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 TIBCO Object Service Broker Installation

This chapter describes how to perform a new installation of the TIBCO Object Service Broker software.

Refer to *TIBCO Object Service Broker Release Notes* for information about migrating from a previous release.

For instructions on installing the TIBCO Object Service Broker UI, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Topics

Overview

The following table gives an overview of the main steps for installing a *new* TIBCO Object Service Broker system. Refer to *TIBCO Object Service Broker Release Notes* for information about migrating from a previous release.

To ensure a successful installation, complete the steps in the order indicated. The base component must be installed before any other component can be installed and used, because it contains routines that are common to and required by other components.

Step	Perform These Tasks
1. Pre-installation tasks	Check Installation Requirements on page 3.
	Uploading the Software on page 7
2. Install and customize the software	Installing the Software on page 9
	Configuring Authorized Libraries on page 22
	See Chapter 2, Customizing the Installation, on page 31
3. Post-installation tasks	See Overview in Chapter 3, Performing Postinstallation Tasks, on page 61.
	See Spin Verification and Cleanup in Chapter 3, Performing Postinstallation Tasks, on page 61.
4. Initialize and test the Native Execution Environment	See Chapter 4, Initializing and Testing the Native Execution Environment, on page 79.
5. Install optional interfaces	See Chapter 5, Installing the Service Gateway for CICS, on page 87.
	See Chapter 6, Installing the Service Gateway for IMS TM, on page 121.
	See Chapter 8, Testing the TIBCO Object Service Broker SDK (C/C++), on page 151.
6. Install and use system management aids	See Chapter 15, TIBCO Object Service Broker Netview Support, on page 253 for information on Netview.
	See *TIBCO Object Service Broker for z/OS Monitoring Performance* for information on the Performance Monitor and other monitoring tools.

Installation Requirements

Hardware Requirements

Processor

To install and operate TIBCO Object Service Broker, your hardware must support z/Architecture.

TIBCO Object Service Broker supports any hardware that runs z/Architecture as specified in *z/Architecture Principles of Operation* (SA22-7832-06). The architectural extensions listed in Chapter 1.2 are not required.

Storage Requirements

TIBCO Object Service Broker requires approximately 1500 cylinders of Direct-Access Storage Device (DASD) space.

Display Devices

TIBCO Object Service Broker supports the use of 3278 and 3279 terminals running in model 2, 3, 4, or 5 mode, and terminals that are compatible with these devices and model types.

Software Requirements

Before installing TIBCO Object Service Broker in a z/OS environment, ensure that the following software components are installed and properly configured.

Operating System

TIBCO Object Service Broker requires z/OS and is supported on 1.9 and above versions.

Language Environment (LE)

TIBCO Object Service Broker requires that IBM's Language Environment product be available during execution. Within a CICS environment, you must install the CICS RDO definitions in CEE.SCEESAMP(CEECCSD) or have the CICS autoinstall be operational for programs.

For routines for which LANGUAGE=LE or LEPERSIST (refer to *TIBCO Object Service Broker for z/OS External Environments*), it is the responsibility of the developer to provide the appropriate CEEUOPT in the LE-compliant external routine. This is because the CEEPIPI enclave is initialized according to the specification of the external routine being called. The same CEEUOPT options must be used for all external routines in each class (LE or LEPERSIST) for the same Execution Environment because the same CEEPIPI enclave cannot support different runtime options.

Optional Supported Software

TCP/IP

TCP/IP communications requires z/OS Communications Server.

Database Server Products

TIBCO Object Service Broker can also access data stored in the following external database server products:

- Adabas
- CA Datacom
- CA IDMS
- DB2
- IMS
- Oracle

Distribution Media and Contents

The most convenient way to obtain a copy of TIBCO Object Service Broker is to download it directly from the TIBCO web site.

Distribution File Format

The files are in a format compatible with IBM System Modification Program/Extended (SMP/E) naming conventions. The product is packaged in SMP/E txlib format.

Component Code

The component code for the TIBCO Object Service Broker product is **S6B**. This component code is registered with the IBM product codes division.

Obtaining the Installation Media

Downloading from the web site

To download TIBCO Object Service Broker from the TIBCO web site, follow these steps:

1. Contact TIBCO Software Inc. for a password and directory information for access to the download web site.
2. Connect to the TIBCO web site at http://download.tibco.com/tibco/ with the required information.
3. Download the software.

Software	Installation Files
TIBCO Object Service Broker	Download the following compressed file: `TIB_osb_6.0.0_zos.zip` The zip file contains these files: – `osb.xml` – `metastr.xml` – `install.bin` – `ostarrec.bin` – `property.bin` – `OSTAREDC`

4. If you use a TIBCO Service Gateway, download the corresponding software.

Software	Installation Files
TIBCO Service Gateway	Download the compressed file applicable to your site: • CICS `TIB_srvcgw-cics_6.0.0_zos.zip` • IMS TM `TIB_srvcgw-imstm_6.0.0_zos.zip` • Files `TIB_srvcgw-file_6.0.0_zos.zip` • DB2 `TIB_srvcgw-db2_6.0.0_zos.zip` • IDMS/DB `TIB_srvcgw-idms_6.0.0_zos.zip` • IMS/DB `TIB_srvcgw-imsdb_6.0.0_zos.zip` • WMQ `TIB_srvcgw-wmq_6.0.0_zos.zip` For details on installing Service Gateway for CICS, Service Gateway for IMS TM, or Service Gateway for WMQ, see Chapter 5, Installing the Service Gateway for CICS, on page 87, Chapter 6, Installing the Service Gateway for IMS TM, on page 121, or Chapter 7, Installing the Service Gateway for WMQ, on page 143. For details on installing the Service Gateway for Files, see the *TIBCO Object Service Broker Managing External Data* manual. For details on installing the other Service Gateways, see the corresponding Service Gateway manual.

Requesting a CD

If you do not want to download the software over the network, you can obtain a CD containing the TIBCO Object Service Broker software. The minimum time period for delivery of a CD is 5 working days. Contact TIBCO Support at

http://support.tibco.com

and request the desired media for your environment. For details, see How to Contact TIBCO Support on page xxi.

Use the TIBCO support web site to check for any LBN bulletins concerning the products you plan to install. The LBNs may advise you of hotfixes that should be installed in conjunction with your installation. The procedure to obtain hotfixes is described elsewhere in this manual. For details, see Hotfixes and Late Breaking News on page 25.

Uploading the Software

If you have acquired TIBCO Object Service Broker by downloading it from the TIBCO web site, you must upload the software to the z/OS host system. Perform the following:

Preparing the Product Files and Utilities for Uploading

1. Download or copy the `TIB_osb_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.
2. Unzip the file to a temporary location on the PC. The file contains the following:
   - `osb.xm1` – a compressed file containing all TIBCO Object Service Broker libraries.
   - `metastr.xm1` – a compressed file containing the TIBCO Object Service Broker MetaStor.
   - `install.bin` – the REXX EXEC to perform the installation.
   - `ostarrec.bin` – the REXX EXEC to uncompress the `.xm1` files.
   - `property.bin` – a template of mandatory install variables required for product installation.
   - `OSTAREDC` – a load module to improve the performance of `OSTAREDC`.
3. Pre-allocate a PDS, fixed block data set on the z/OS host system with the following name:

   ```
   <HLQ>.INSTALL
   ```

   where <HLQ> is any valid high-level qualifier and may consist of multiple levels, such as ABCD.XYZ. Note that this <HLQ> will be used during the installation. See the sample JCL in the Step 4.
4. Pre-allocate the following sequential data sets on the z/OS host system:

   ```
   <HLQ>.OSB.XM1                    (size 77,570 KB)
   <HLQ>.METASTR.XM1                (size 10,715 KB)
   ```

   Use the same <HLQ> as the previous data set. Below is sample JCL to allocate these data sets. Provide a JOB card and submit the JCL.

   ```
   //ALLOC EXEC PGM=IEFBR14
   //DD1   DD DSN=<HLQ>.INSTALL,
   //       DISP=(,CATLG,DELETE),UNIT=SYSDA,
   //       DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
   //       SPACE=(TRK,(5,15,100))
   //DD2   DD DSN=<HLQ>.OSB.XM1,
   //       DISP=(,CATLG,DELETE),UNIT=SYSDA,
   ```

```
//       DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
//       SPACE=(TRK,(2000,50))
//DD3    DD DSN=<HLQ>.METASTR.XM1,
//       DISP=(,CATLG,DELETE),UNIT=SYSDA,
//       DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
//       SPACE=(TRK,(250,10))
```

5. FTP `install.bin`, `ostarrec.bin`, and `property.bin` to your z/OS system in BIN mode to the `<HLQ>.INSTALL` data set. Name these utilities `INSTALL`, `OSTARREC`, and `PROPERTY`, respectively.
6. FTP the `osb.xm1` file in BIN mode to the `<HLQ>.OSB.XM1` data set.
7. FTP the metastr`.xm1` file in BIN mode to the `<HLQ>.METASTR.XM1` data set.

Installing the OSTAREDC Program

1. FTP the `OSTAREDC` file to z/OS in BIN mode to a sequential data set with LRECL=80 and RECFM=FB.
2. In ISPF 3.4, against this data set, type the following:

   `"RECEIVE INDA(/)"`

   When prompted, specify `DA('<HLQ>.INSTLOAD')` as the name of the load library where you want the `OSTAREDC` program restored.
3. Edit `OSTARREC` as follows:

   — Issue the command `"FIND OSTAREDC 1"`.

   — Change the constant after the equal sign to contain the full data set name of the program. The string must start with a double quote and a single quote, and end with a single quote and a double quote (the double quotes delimit the string and the single quotes tell TSO that the data set name is fully qualified). For example, change the following:

   `OSTAREDC = "'<HLQ>.INSTLOAD(OSTAREDC)'"`

   to

   `OSTAREDC = "'`*your.load.library*`(OSTAREDC)'"`

   where *your.load.library* is the name of the library referenced in Step 2.

For additional information, see Data Transmission for z/OS, page 449.

Installing the Software

You must perform the installation under an ISPF environment.

These instructions are for a new TIBCO Object Service Broker installation; to migrate from a previous version, see *TIBCO Object Service Broker Release Notes*.

You can start the installation if you have uploaded the following data sets as described in the Uploading the Software on page 7:

- `<HLQ>.INSTALL`
- `<HLQ>.OSB.XM1`
- `<HLQ>.METASTR.XM1`

The <HLQ> referenced in this chapter is the high-level qualifier you specified when you uploaded the product software. This is the value of the INSTALL and XM1 files you specified. It is used as the default value for all distribution files created when an XM1 is uncompressed. An ISPF edit macro called OSEMOD is supplied for the purpose of customizing installed data sets. It will be used later in the installation process. It is equivalent to the value of symbolic parameter HLQ as described in OSEMOD (see Required Variables on page 395).

Automated versus Manual Installation

Use the automated installation for a minimal installation; for example, for a proof of concept. Use the manual installation to customize the product to your site's requirements.

Installation Overview

To install TIBCO Object Service Broker, perform the following:

1. Determine your system environment values listed in Table 3, Subset of OSEMOD Variables, on page 10.
2. Edit the Properties File using the values determined in Step 1.
3. Prepare the Authorized Environment.
4. Install the software:
   - For the automated installation, see Automated Installation on page 13.
   - For the manual installation, see Manual Installation on page 18.

5. Perform Installation Verification Procedure on page 16.

System Environment Checklist

Before you begin the installation, review the system environment information described in Table 3 and determine whether you will use the default value or provide your own value. See Base Installation Listing on page 27 for a list of the data set structure.

Table 3 Subset of OSEMOD Variables

Description	OSEMOD Variable	Default Value	Your Value
High level qualifier for uploaded data sets INSTALL, OSB.XM1, and METASTR.XM1.	HLQ	Specified on upload	
High level qualifier for non-VSAM and VSAM data sets you are authorized to create.	$HLQNONV$ $HLQVSAM$	TIBCO.TESTNV TIBCO.TESTVS	
Second level qualifier for install files.	$INSTVER$	INS60	
Second level qualifier for TIBCO Object Service Broker system files.	SLQ	OSB60	
Second level qualifier for SMP/E libraries	SMP	SMP60	
For SMS Shops – managementclass, dataclass and storageclass, if required			
For new non-VSAM data sets.	$NMGTCLAS $NDATCLAS$ $NSTOCLAS$	STANDARD STANDARD S6BNONV	
For new VSAM data sets.	$VMGTCLAS $VDATCLAS$ $VSTOCLAS$	STANDARD STANDARD S6BVSAM	
High level qualifier of Language Environment libraries for SCEELKED and SCEEBIND.	$CEELIB$	CEE	

Table 3 Subset of OSEMOD Variables

Description	OSEMOD Variable	Default Value	Your Value
High level qualifier of IBM's Callable Services library CSSLIB.	$CSSLIB$	SYS1	
Data Object Broker communication ID (if cross-memory facility will not be used).	TDS	OSBDOB	

For additional information, see Required Variables on page 395.

Edit the Properties File

Use the PROPERTY member in `<HLQ>.INSTALL` as a template, and modify to suit your requirements. Table 4 describes keywords in the properties file that correspond to the system environment variables in System Environment Checklist.

Table 4 Properties File Keywords

Keyword	Description
`INSTALL=`	To install the TIBCO Object Service Broker base component, specify `OSB`: `INSTALL=OSB`
`HLQNONV=`	High level qualifier for non-VSAM data sets.
`HLQVSAM=`	High level qualifier for VSAM data sets.
`INSTVER=`	Second level qualifier for install files.
`SLQ=`	Second level qualifier for TIBCO Object Service Broker system files.
`SMP=`	Second level qualifier for SMP/E libraries.
`SMS=`	`YES` for SMS site, `NO` for non-SMS site. **Warning:** If you select the `SMS=YES` option, be sure to specify SMS-managed data-set names. The SMS automatic class selection (ACS) rules for your site determine whether a data-set name is eligible for SMS management. The name you specified that is determined by the ACS routines to be SMS-eligible is SMS-managed. Otherwise, the result is unpredictable.

Table 4 Properties File Keywords

Keyword	Description
COMPAT=	Use if SMS=YES. Valid values: YES for SMS compatible data set name classes; NO for SMS non-compatible data set name classes. If COMPAT=NO, specify the following: • NMGTCLAS – MANAGEMENTCLASS for non-VSAM data sets • NDATCLAS – DATACLASS for non-VSAM data sets • NSTOCLAS – STORAGECLASS for non-VSAM data sets • VMGTCLAS – MANAGEMENTCLASS for VSAM data sets • VDATCLAS – DATACLASS for VSAM data sets • VSTOCLAS – STORAGECLASS for VSAM data sets
VOLSER=	If SMS=YES, specify one DASD volume for VSAM data set allocation. Default is USER01. If SMS=NO, specify three DASD volumes separated by commas. Defaults are OSBS06, OSBD18, OSBB02. • vol1 – DASD volser for temp work files • vol2 – DASD volser for install files • vol3 – DASD volser for TIBCO Object Service Broker system files
CEELIB=	High level qualifier of Language Environment libraries.
CSSLIB=	High level qualifier of IBM's Callable Services library CSSLIB.
TDS=	Data Object Broker communication ID (if cross-memory facility will not be used).

Prepare the Authorized Environment

Authorize the <HLQNONV>.<INSTVER>.AUTH library that will be created during the installation.

Automated Installation

To exit the interactive session at any time after executing the REXX exec INSTALL, do the following:

1. Press `PA1`
2. Enter `hi`
3. Press `ENTER` twice

STEP 1: **Execute File Tailoring EXEC to start installation.**

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

STEP 1 will verify that files can be allocated successfully using the values provided in the PROPERTY file. Test files of type sequential, PDS, PDSE, and VSAM will be allocated then deleted. Installation will stop if any test allocation fails. You should investigate the cause, correct the condition and repeat STEP 1.

The OSB.JCL data set is created at the successful completion of this step.

STEP 2: **Edit the Job card to your site's standards and run Job OSB.JCL**

JCL in: `<HLQ>.OSB.JCL`

Data Set: `<HLQ>.OSB.JCL (SUB data set)`

This batch job will uncompress the OSB.XM1 and METASTR.XM1 files to produce the distribution libraries.

If you modify the job name, make sure it does not exceed seven characters.The job should successfully complete with a return code of 0.

STEP 3: **Edit OSEMOD. (Optional)**

If you wish to make additional changes to the values of OSEMOD variables, make the changes now.

OSEMOD is an ISPF edit macro that is used to edit TIBCO-supplied data sets. Variables in the data sets are changed during the next step of the installation process. A complete list of variables may be found at Required Variables on page 395. It is suggested that values be supplied for `$HOST$`, **`$DOBPORT$`** and `$EEPORT$` to take advantage of TCP/IP.

Member in: `<HLQ>.FILECLS`

Member: `OSEMOD`

STEP 4: **Create and customize work copies of data sets.**

Member in: `<HLQ>.OSB.FILEI`

Member: `S6A1CUST (EX member)`

The following work copies are created and customized with values specified by OSEMOD variables:

`Customized copy – Library Description`

- `<HLQNONV>.<INSTVER>.CLIST – CLIST`
- `<HLQNONV>.<INSTVER>.CNTL – CNTL`
- `<HLQNONV>.<INSTVER>.JCL – Sample JCL`
- `<HLQNONV>.<INSTVER>.OSB.JOBS – Install jobs for new OSB system`

STEP 5: **Modify STATUS of installation jobs, as required.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `JOBSA (EDIT member)`

Member JOBSA contains the list of jobs that will be submitted in the next step. Each non-commented line in the data set contains three fields: name of the member to be submitted, highest expected return code and the job status.

Lines in JOBSA member are evaluated in the order they are listed and the STATUS field indicates if the corresponding job will be submitted. Also, the next job can be submitted only if the previous one completed with its expected return code.

Valid status: INSTALL (run the job), FUTURE/OPTIONAL (skip the job), DONE (job already completed).

By editing STATUS fields in JOBSA member, you control which steps will be initiated in step 6.

Upon completion of step 5 the JOBSA member STATUS fields are updated to reflect the outcome of each job.

Status is modified from INSTALL to DONE if the job completion code was equal or less than the stated return code.

If any of the jobs in the list fails to complete with the expected return code, the jobs that follow will not be executed, but the STATUS field of those which preceded it on the list will be updated with STATUS DONE. Therefore, you can correct the reason for failure and simply re-run the step 5.

You can modify the STATUS of any job as per your requirement. For example, if your shop normally ACCEPTs the product FMID at some future time, then change the status of S6A4ACPT from INSTALL to FUTURE. If you wish to install any of the local Service Gateways later, it is mandatory that ACCEPT is done first. Note that you must ACCEPT the base OSB FMID before applying any hotfix maintenance using SMP/E.

Status is modified from INSTALL to DONE only if the job's completion code is equal to or less than the stated return code.

Note: Do not edit member JOBSA once the install jobs are initiated in STEP 6.

You can keep the member unchanged if the STATUS fields of all jobs is acceptable (it will initiate all the jobs required).

STEP 6: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A2RUNJ (EX member)`

SEND messages are directed to the userid specified in the NOTIFY parameter of each job submitted, informing user of submission and normal completion or abnormal termination. The successful completion of the final job in JOBSA list is accompanied by the message ALL MEMBERS PROCESSED.

This completes the auto-installation process. For a list of installed libraries and data sets, see Base Installation Listing on page 27.

STEP 7: **Install the optional Service Gateways.**

For installation details for the Service Gateways for CICS, IMS TM, and WMQ, refer to Chapter 5, Installing the Service Gateway for CICS, Chapter 6, Installing the Service Gateway for IMS TM, and Chapter 7, Installing the Service Gateway for WMQ. For the Service Gateway for Files, refer to *TIBCO Object Service Broker Managing External Data*. For all other gateways, refer to the gateway specific manuals.

STEP 8: **Apply the latest hotfix level.**

Use the LBN system to determine the latest hotfix level.

Download it and apply any required SMP maintenance at this point as per the documentation with the hotfix.

STEP 9: **Start the Data Object Broker.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A8DOB  (Edit JOBCARD and SUB member)`

You must respond to a WTOR on this occasion to allow the Data Object Broker to start. For example:

```
12.02.21 JOB04800 S6BKR006A-XYZ REDOLOG AND CACHE
FORMATTED. REPLY TO S6BKR098
12.02.21 JOB04800 *454 S6BKR098-XYZ READ S6BKR006 AND
REPLY G TO GO, C TO CANCEL
12.02.43 JOB04800 R 454,G
```

STEP 10: **Apply database maintenance supplied with the latest hotfix level.**

If there is any database maintenance supplied with the hotfix downloaded in Step 8, apply it now as instructed in the hotfix documentation.

Perform Installation Verification Procedure

Perform the IVP as follows:

1. Logon to the Data Object Broker. Under ISPF option 6, do the following:

   Type the following to allocate the CLIST library:

   ```
   altlib activate application(clist)
      da('<HLQNONVS>.<INSTVER>.CLIST')
   ```

Type the following to logon to the Data Object Broker:

```
user u(sysadmin) p(sysadmin)
```

The TIBCO Object Service Broker work screen displays as user SYSADMIN.

2. On the work screen, move the cursor to the command line and type the following:

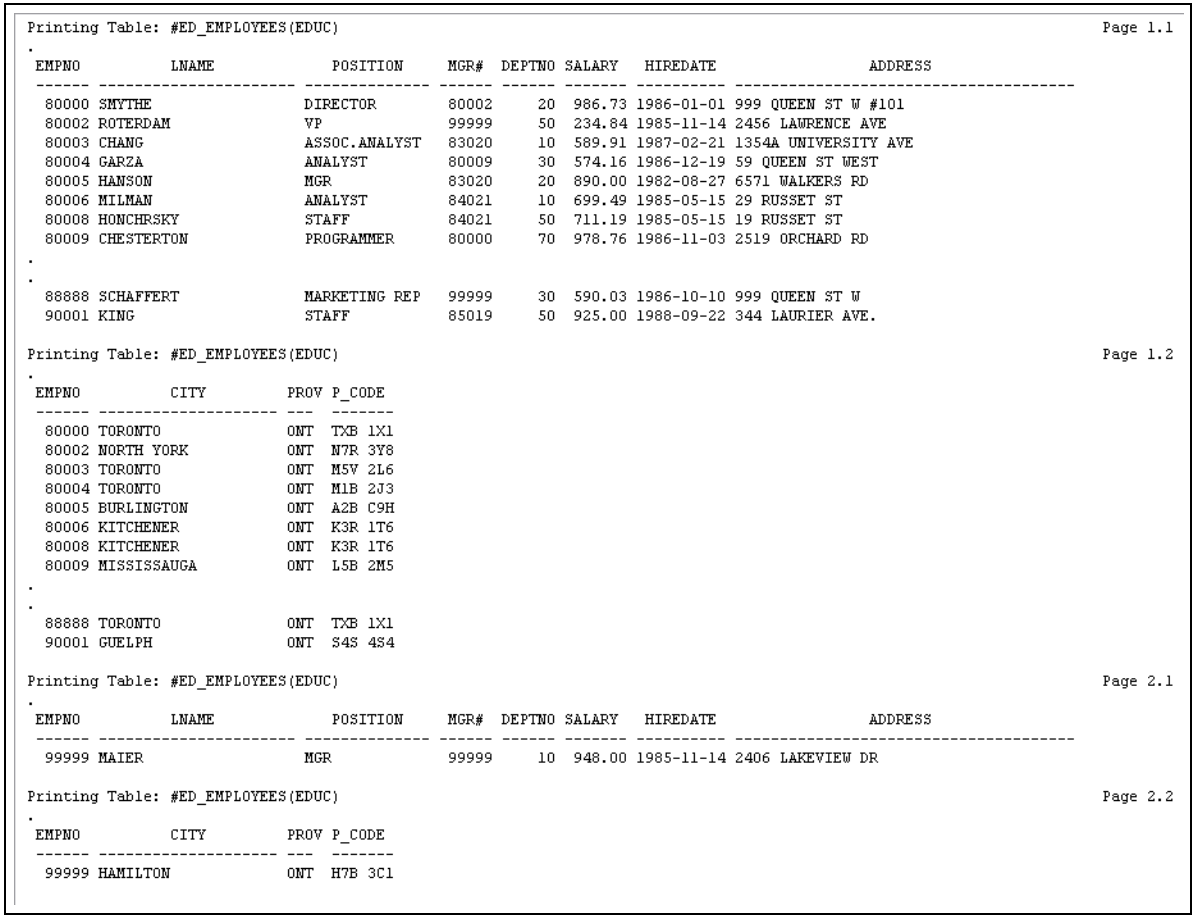
```
COMMAND ==> ex tableprint('#ed_employees(educ)')
```

Following message appears at the bottom of the screen:

12:31:39 Printed #ED_EMPLOYEES(EDUC)

3. To validate output in SDSF, browse the job name of your TSO session and select the DDNAME SYSnnnnn with job class A and destination LOCAL.

Output should appear as below:



```
Printing Table: #ED_EMPLOYEES(EDUC)                                                                                    Page 1.1
.
 EMPNO          LNAME              POSITION     MGR#  DEPTNO SALARY   HIREDATE                ADDRESS
 ------ -------------------- -------------- ------ ------ ------- ---------- ----------------------------------------
  80000 SMYTHE               DIRECTOR       80002     20  986.73 1986-01-01 999 QUEEN ST W #101
  80002 ROTERDAM             VP             99999     50  234.84 1985-11-14 2456 LAWRENCE AVE
  80003 CHANG                ASSOC.ANALYST  83020     10  589.91 1987-02-21 1354A UNIVERSITY AVE
  80004 GARZA                ANALYST        80009     30  574.16 1986-12-19 59 QUEEN ST WEST
  80005 HANSON               MGR            83020     20  890.00 1982-08-27 6571 WALKERS RD
  80006 MILMAN               ANALYST        84021     10  699.49 1985-05-15 29 RUSSET ST
  80008 HONCHRSKY            STAFF          84021     50  711.19 1985-05-15 19 RUSSET ST
  80009 CHESTERTON           PROGRAMMER     80000     70  978.76 1986-11-03 2519 ORCHARD RD
.
.
  88888 SCHAFFERT            MARKETING REP  99999     30  590.03 1986-10-10 999 QUEEN ST W
  90001 KING                 STAFF          85019     50  925.00 1988-09-22 344 LAURIER AVE.

Printing Table: #ED_EMPLOYEES(EDUC)                                                                                    Page 1.2
.
 EMPNO          CITY          PROV P_CODE
 ------ -------------------- ---  -------
  80000 TORONTO              ONT  TXB 1X1
  80002 NORTH YORK           ONT  N7R 3Y8
  80003 TORONTO              ONT  M5V 2L6
  80004 TORONTO              ONT  M1B 2J3
  80005 BURLINGTON           ONT  A2B C9H
  80006 KITCHENER            ONT  K3R 1T6
  80008 KITCHENER            ONT  K3R 1T6
  80009 MISSISSAUGA          ONT  L5B 2M5
.
.
  88888 TORONTO              ONT  TXB 1X1
  90001 GUELPH               ONT  S4S 4S4

Printing Table: #ED_EMPLOYEES(EDUC)                                                                                    Page 2.1
.
 EMPNO          LNAME              POSITION     MGR#  DEPTNO SALARY   HIREDATE                ADDRESS
 ------ -------------------- -------------- ------ ------ ------- ---------- ----------------------------------------
  99999 MAIER                MGR            99999     10  948.00 1985-11-14 2406 LAKEVIEW DR

Printing Table: #ED_EMPLOYEES(EDUC)                                                                                    Page 2.2
.
 EMPNO          CITY          PROV P_CODE
 ------ -------------------- ---  -------
  99999 HAMILTON             ONT  H7B 3C1
```

Manual Installation

To exit the interactive session at any time after executing the REXX exec INSTALL, do the following:

1. Press `PA1`
2. Enter `hi`
3. Press `ENTER` twice

STEP 1: **Execute File Tailoring EXEC to start installation.**

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

STEP 1 will verify that files can be allocated successfully using the values provided in the PROPERTY file. Test files of type sequential, PDS, PDSE, and VSAM will be allocated then deleted. Installation will stop if any test allocation fails. You should investigate the cause, correct the condition and repeat STEP 1.

The OSB.JCL data set is created at the successful completion of this step.

STEP 2: **Edit the Job card to your site's standards and run Job OSB.JCL**

JCL in: `<HLQ>.OSB.JCL`

Data Set: `<HLQ>.OSB.JCL (SUB data set)`

This batch job will uncompress the OSB.XM1 and METASTR.XM1 files to produce the distribution libraries.

If you modify the job name, make sure it does not exceed seven characters.The job should successfully complete with a return code of 0.

STEP3: **Edit OSEMOD. (Optional)**

If you wish to make additional changes to the values of OSEMOD variables, make the changes now.

Member in: `<HLQ>.FILECLS`

Member: `OSEMOD`

STEP 4: **Create work copies of the CLIST, CNTL, JCL and OSB.JOBS data sets.**

Below is a sample JCL to allocate and propagate the work copies. Provide a JOB card and submit the JCL.

```
//ALLOC  EXEC PGM=IEBCOPY
//ICLIST   DD DISP=SHR,DSN=<HLQ>.FILECLS
//OCLIST   DD DISP=(NEW,CATLG,DELETE),
//         DSN=<HLQNONV>.<INSTVER>.CLIST,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         SPACE=(TRK,(5,10,5),RLSE),UNIT=SYSDA
//ICNTL    DD DISP=SHR,DSN=<HLQ>.FILECTL
//OCNTL    DD DISP=(NEW,CATLG,DELETE),
//         DSN=<HLQNONV>.<INSTVER>.CNTL,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         SPACE=(TRK,(10,10,25),RLSE),UNIT=SYSDA
//IJCL     DD DISP=SHR,DSN=<HLQ>.FILEJCL
//OJCL     DD DISP=(NEW,CATLG,DELETE),
//         DSN=<HLQNONV>.<INSTVER>.JCL,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         SPACE=(CYL,(2,5,40),RLSE),UNIT=SYSDA
//IJOBS    DD DISP=SHR,DSN=<HLQ>.OSB.FILEI
//OJOBS    DD DISP=(NEW,CATLG,DELETE),
//         DSN=<HLQNONV>.<INSTVER>.OSB.JOBS,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         SPACE=(TRK,(5,10,5),RLSE),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 COPY INDD=ICLIST,OUTDD=OCLIST
 COPY INDD=ICNTL,OUTDD=OCNTL
 COPY INDD=IJCL,OUTDD=OJCL
 COPY INDD=IJOBS,OUTDD=OJOBS
//
```

STEP 5: **Customize members of CNTL data set.**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `DPLXREDO (EDIT member, invoke OSEMOD in command line)`

CNTL members to edit with OSEMOD: `DPLXREDO`, `OSRUNPRM`, `PARMBAT`, `PARMCICS`, `PARMDC`, `PARMNEE`, `PARMTSO`, `RELAYCFG`, `SPLXJRN1`, `SPLXJRN2`, `SPLXREDO`, `SPLXSEG0`, `SPLXSEG1`, `SPLXSG99`

Note: Be sure to edit with OSEMOD any member of the working copies of CLIST and JCL that you plan to execute or run. Also check any reference to CNTL parameters in the JCL and edit the affected CNTL member with OSEMOD if required.

Invoking OSEMOD in the command line – If you have not added the `<HLQNONV>.<INSTVER>.CLIST` library in your installation's SYSPROC or SYSEXEC concatenation, you can manually type in the command below from TSO/ISPF Option 6, TSO Commands. It must be reentered every time you get into ISPF, and must be entered on each half of an ISPF split screen. Prefix it with TSO space if you enter it on an ISPF screen other than Option 6. Do not forget to edit CLIST members ADMIN and USER with OSEMOD.

```
ALTLIB ACTIVATE APPLICATION(CLIST)
  DA('<HLQNONV>.<INSTVER>.CLIST')
```

STEP 6: **Individually customize and submit install job.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A3ALOC (EDIT member, invoke OSEMOD in command line, then SUB member)`

Repeat for the following jobs: `S6A3APLY`, `S6A4ACPT`, `S6A5CFGR`, `S6A5DBDG`, `S6A5FRMT`, `S6A6POST`, `S6A7BKUP`.

See member JOBSA for the expected return codes.

For a list of installed libraries and data sets, see Base Installation Listing on page 27.

STEP 7: **Install the optional Service Gateways.**

For installation details for the Service Gateways for CICS, IMS TM, and WMQ, refer to Chapter 5, Installing the Service Gateway for CICS, Chapter 6, Installing the Service Gateway for IMS TM, and Chapter 7, Installing the Service Gateway for WMQ. For the Service Gateway for Files, refer to *TIBCO Object Service Broker Managing External Data*. For all other gateways, refer to the gateway specific manuals.

STEP 8: **Apply the latest hotfix level.**

Use the LBN system to determine the latest hotfix level.

Download it and apply any required SMP maintenance at this point as per the documentation with the hotfix.

STEP 9: **Start the Data Object Broker.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A8DOB (Edit JOBCARD and SUB member)`

You must respond to a WTOR on this occasion to allow the Data Object Broker to start. For example:

```
12.02.21 JOB04800 S6BKR006A-XYZ REDOLOG AND CACHE
FORMATTED. REPLY TO S6BKR098
12.02.21 JOB04800 *454 S6BKR098-XYZ READ S6BKR006 AND
REPLY G TO GO, C TO CANCEL
12.02.43 JOB04800 R 454,G
```

STEP 10: **Apply the latest hotfix level.**

If there is any database maintenance supplied with the hotfix downloaded in Step 8, apply it now as instructed in the hotfix documentation.

Configuring Authorized Libraries

Perform the following tasks to configure authorized libraries:

Task A Update IEAAPFxx in SYS1.PARMLIB

Update the IEAAPF*xx* member in the SYS1.PARMLIB data set to include the name of the TIBCO Object Service Broker load library to be APF authorized. If you want to maintain an authorized DD concatenation, any other load library that is concatenated and accessed from the same address space with the TIBCO Object Service Broker library also must be added to the IEAAPF*xx* member (for example, the load library specified in the EXLIB statement in the USER EXEC, if used).

Task B Modify Security Rules

If necessary, modify security rules (such as CA-ACF2, RACF, CA-TOP SECRET, and so on) to make the data set available in update mode to the installer and enable the specification of an authority code of 1 on linkage editor control statements.

Task C Update IKJTSOxx in SYS1.PARMLIB

Update the IKJTSO*xx* member in the SYS1.PARMLIB data set to include programs executed in TSO that use XMS functions, specifying that they are to be executed as authorized programs. The following programs must be added to the AUTHPGM section of IKJTSO*xx* by your systems programmer:

- S6BTSO
- S6BCR000
- S6BTLADM
- S6BBATCH

Task D Add CLIST Library to SYSPROC or SYSEXEC

To allow TIBCO Object Service Broker 6.0 EXECs/CLISTs to be invoked under TSO without having to specify a library name, add the <HLQNONV>.<INSTVER>.CLIST library in your installation's SYSPROC or SYSEXEC concatenation.

Reinstalling the Software

If you have a data set allocation failure, an incorrect library specification, SMP/E related problems with the APPLY job S6A3APLY, or other problems, you may need to reinstall the software. To do so, perform the following:

STEP 1: **Correct the problem that caused the failure or malfunction.**

STEP 2: **Run the OSB cleanup job.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A9CLEN  (Edit JOBCARD and SUB member)`

Job should complete with a return code of 0.

STEP 3: **Change the job status.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `JOBSA (Change any job with the status DONE to INSTALL)`

STEP 4: **Restart jobs.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A2RUNJ  (EX member)`

Uninstalling the Software

To uninstall the product, perform the following:

STEP 1: **Run the OSB cleanup job.**

Member in: `<HLQNONV>.<INSTVER>.OSB.JOBS`

Member: `S6A9CLEN  (Edit JOBCARD and SUB member)`

STEP 2: **Manually delete the following data sets (in the specified sequence):**

1. `$HLQ$.ASM`
2. `$HLQ$.COBOL`
3. `$HLQ$.FILECLS`
4. `$HLQ$.FILECTL`
5. `$HLQ$.FILEEM1`
6. `$HLQ$.FILEEM2`
7. `$HLQ$.FILEJCL`
8. `$HLQ$.FILEMIG`
9. `$HLQ$.FILEOBJ`
10. `$HLQ$.FILETRK`
11. `$HLQ$.FILEUP4`
12. `$HLQ$.FILEUP5`
13. `$HLQ$.FILEXML`
14. `$HLQ$.MACRO`
15. `$HLQ$.OSB.FILEI`
16. `$HLQ$.UNICODE`
17. `$HLQ$.METASTR1`
18. `$HLQ$.METASTR2`
19. `$HLQ$.METASTR3`
20. `$HLQNONV$.$INSTVER$.CLIST`
21. `$HLQNONV$.$INSTVER$.CNTL`
22. `$HLQNONV$.$INSTVER$.JCL`
23. `$HLQNONV$.$INSTVER$.MIGRATE`
24. `$HLQNONV$.$INSTVER$.OSB.JOBS`

At this point, you should only have the uploaded data sets:

- `$HLQ$.INSTALL`
- `$HLQ$.OSB.XM1`
- `$HLQ$.METASTR.XM1`

Manually delete the data sets.

Hotfixes and Late Breaking News

TIBCO Object Service Broker maintenance is normally implemented by upgrading to the latest product release. For Severity 1 or 2 problems, a fix may be created specifically for that issue at your current software level. Such fixes will be delivered to you in a package called a *hotfix*.

Hotfixes

Hotfixes are generally cumulative so that any preceding hotfix maintenance will be included in the latest hotfix package. Hotfixes are numbered consecutively within each product/release combination.

You will be able to download the hotfix from your directory at the following site:

ftp://support-ftp.tibco.com

The hotfix package will need to be unzipped and then placed into data sets on your z/OS system. When FTPing fixes, always use BINARY FTP mode with no CR/LF options. z/OS hotfix data sets will need to be unwrapped using the standard TIBCO Object Service Broker tool OSTARREC.

Depending on the .XM suffix of the hotfix, the target data set should have the following characteristics in order for the OSTARREC exec to unwrap the data set successfully:

`.XMT -RECFM=FB LRECL=88 BLKSIZE=n*88 /*` wrapped with data integrity `*/`

`.XM1 -RECFM=FB LRECL=1024 BLKSIZE=n*1024 /*` wrapped with data integrity and compressed with IBM's TRSMAIN utility `*/`

`.XM2 -RECFM=FB LRECL=1024 BLKSIZE=n*1024 /*` compressed with IBM's TRSMAIN utility only `*/`

During the OSTARREC process, you can supply a target data set name by typing for example, DSNAME('OSB.IMSDB.HFX01.JOBS') when prompted; otherwise, your TSO prefix will be used as the high level qualifier for the hotfix data set.

All z/OS program maintenance will be applied using SMP/E. Rule and table maintenance will be applied using TIBCO Object Service Broker tools. Detailed instructions for application of hotfixes will be included in the hotfix package and optionally supplemented by comments added to your problem Service Request by TIBCO Support.

Using SMP/E, you should ACCEPT the FMID and any previous hotfixes before executing the APPLY for new hotfix maintenance.

Late Breaking News (LBN)

Customers can register for automatic LBN email notifications for each installed product by logging on to the TIBCO Support web site. You can also search and view LBNs manually. LBNs will inform you of the latest product advisories since the product shipped.

Base Installation Listing

A base installation of TIBCO Object Service Broker creates a minimal, implementable system; this system consists of libraries and data sets that will support the startup of a Data Object Broker and Execution Environments using TSO and Native. A working TIBCO Object Service Broker system is made up of essential and support libraries and data sets. A base installation also creates an SMP/E environment that is required to support future maintenance of the product and its optional, dependent components.

Essential Libraries and Data Sets

Purpose	Libraries / Data Sets
TIBCO Object Service Broker authorized load library	$HLQNONV$.$INSTVER$.AUTH.
Data Object Broker initialization parameter OSRUNPRM and Execution Environment configuration members PARMBAT, PARMNEE, and PARMTSO	$HLQNONV$.$INSTVER$.CNTL
Database definition library	$HLQVSAM$.SLQ.DBDLIB $HLQVSAM$.SLQ.DBDLIB.DATA
Segment 0 – TIBCO Object Service Broker MetaStor with three page data sets	$HLQVSAM$.SLQ.$SEG0NAM$.PAGE1 $HLQVSAM$.SLQ.$SEG0NAM$.PAGE1.DATA $HLQVSAM$.SLQ.$SEG0NAM$.PAGE2 $HLQVSAM$.SLQ.$SEG0NAM$.PAGE2.DATA HLQVSAM$.$SLQ$.$SEG0NAM$.PAGE3 $HLQVSAM$.SLQ.$SEG0NAM$.PAGE3.DATA
Segment 1 – User data segment with three page data sets	$HLQVSAM$.SLQ.$SEG1NAM$.PAGE1 $HLQVSAM$.SLQ.$SEG1NAM$.PAGE1.DATA $HLQVSAM$.SLQ.$SEG1NAM$.PAGE2 $HLQVSAM$.SLQ.$SEG1NAM$.PAGE2.DATA $HLQVSAM$.SLQ.$SEG1NAM$.PAGE3 $HLQVSAM$.SLQ.$SEG1NAM$.PAGE3.DATA
Segment 99 – Security segment with a single page data set	$HLQVSAM$.SLQ.$SEG99NAM$.PAGE1 $HLQVSAM$.SLQ.$SEG99NAM$.PAGE1.DATA

Purpose	Libraries / Data Sets
Two caches	$HLQNONV$.SLQ.CACHE1 $HLQNONV$.SLQ.CACHE2
Redolog pending	$HLQNONV$.SLQ.REDOLOG.PENDING
Archlog	$HLQNONV$.SLQ.ARCHLOG
Two journals	$HLQVSAM$.SLQ.JRNL1 $HLQVSAM$.SLQ.JRNL1.DATA $HLQVSAM$.SLQ.JRNL2 $HLQVSAM$.SLQ.JRNL2.DATA
Redolog	$HLQVSAM$.SLQ.REDOLOG $HLQVSAM$.SLQ.REDOLOG.DATA
Resource	$HLQVSAM$.SLQ.RESOURCE $HLQVSAM$.SLQ.RESOURCE.DATA $HLQVSAM$.SLQ.RESOURCE.INDEX
CLIST members USER & ADMIN	$HLQNONV$.$INSTVER$.CLIST
Sample JCL library	$HLQNONV$.$INSTVER$.JCL
Miscellaneous	$HLQNONV$.SLQ.DOBDUMP $HLQNONV$.SLQ.EEDUMP $HLQNONV$.SLQ.MSWDUMP $HLQNONV$.SLQ.MODEL.DSCB

Support Libraries and Data Sets

Purpose	Libraries / Data Sets
System backup generation data group	$HLQNONV$.SLQ.BACKUP
Journal spin archive generation data group	$HLQNONV$.SLQ.JOURNAL.SPINOUT
Journal spin merge archive generation data group	$HLQNONV$.SLQ.JOURNAL.SPINMRG
Relay configuration file	$HLQNONV$.SLQ.RELAYCFG

Purpose	Libraries / Data Sets
Segment 0 pointer – Check report and data files	$HLQNONV$.SLQ.$SEG0NAM$.ERRLOG $HLQNONV$.SLQ.$SEG0NAM$.ORPHAN $HLQNONV$.SLQ.$SEG0NAM$.REFLOG
Segment 1 pointer – Check report and data files	$HLQNONV$.SLQ.$SEG1NAM$.ERRLOG $HLQNONV$.SLQ.$SEG1NAM$.ORPHAN $HLQNONV$.SLQ.$SEG1NAM$.REFLOG
Segment 99 pointer – Check report and data files	$HLQNONV$.SLQ.$SEG99NAM$.ERRLOG $HLQNONV$.SLQ.$SEG99NAM$.ORPHAN $HLQNONV$.SLQ.$SEG99NAM$.REFLOG

SMP/E environment

The following SMP/E libraries are allocated and utilized during the installation:

Purpose	Libraries
Global CSI and support files	$HLQVSAM$.SMP.GLOBAL.CSI $HLQVSAM$.SMP.GLOBAL.DATA $HLQVSAM$.SMP.GLOBAL.INDEX $HLQNONV$.SMP.LOG $HLQNONV$.SMP.LOGA $HLQNONV$.SMP.LTS $HLQNONV$.SMP.MTS $HLQNONV$.SMP.PTS $HLQNONV$.SMP.SCDS $HLQNONV$.SMP.STS
Target libraries	$HLQNONV$.SMP.C – storage for ++DATA C code members $HLQNONV$.SMP.CLST – storage for ++CLIST members $HLQNONV$.SMP.H – storage for ++DATA H(header) code members $HLQNONV$.SMP.LOAD – target LOAD library $HLQNONV$.SMP.MAC – storage for macros $HLQNONV$.SMP.SAMP – storage for all ++SAMP members except ASM and COBOL $HLQNONV$.SMP.SRCSAMP – storage for ASM and COBOL source code

Purpose	Libraries
Distribution libraries	$HLQNONV$.SMP.AC– storage for ++DATA C code members $HLQNONV$.SMP.ACLST – storage for ++CLIST members $HLQNONV$.SMP.AH – storage for ++DATA H(header) code members $HLQNONV$.SMP.ALOAD – distribution ALOAD library $HLQNONV$.SMP.AMAC – storage for macros $HLQNONV$.SMP.ASAMP – storage for all ++SAMP members except ASM and COBOL $HLQNONV$.SMP.ASRCSAMP – storage for ASM and COBOL source code

Chapter 2 Customizing the Installation

This chapter describes how to customize your TIBCO Object Service Broker installation.

Topics

Overview

The following are steps for customizing a *new* TIBCO Object Service Broker system. You should complete the steps in the order indicated.

Modifying Journal and Segment Definitions

Members SPLXJRN1, SPLXJRN2, SPLXREDO, SPLXSEG0, SPLXSEG1 and SPLXSG99 in the CNTL data set contain statements for use by IDCAMS to allocate Data Object Broker data sets. Modify these members to your new values and use them as SYSIN input to an IDCAMS job. Refer to the corresponding members in the CNTL distribution source library `<HLQ>.FILECTL` for non-customized references to the variables listed below.

Member DBJCL in the JCL data set is the job that generates TIBCO Object Service Broker's database definitions. Refer to the corresponding member DBJCL in the distribution source library `<HLQ>.FILEJCL` for non-customized references to the variables listed below. It contains references to variables that are used to generate segment, journal, cache, redolog, archlog and resource names. Modify member DBJCL in the JCL data set to your new values and submit the job.

For more details on customizing definitions, see Appendix A, Installation Variables, on page 393.

Variable	Description	Default Value	Your Value
$JRNLSIZ$	Journal size	25 cylinders	
$REDOSIZ$	Redolog size	50 cylinders	
$REDODUP$	Duplex redolog? If Y, specify DPLXREDO as value of $REDOVDF$	N	
$REDOVDF$	CNTL member for IDCAMS	SPLXREDO	
$SEG0NAM$	MetaStor segment name	SEG00	
$SEG0SIZ$	MetaStor segment size	100 cylinders	
$SEG1NAM$	Customer segment name	SEG01	
$SEG1SIZ$	Customer segment size	25 cylinders	
$SEG99NAM$	Audit security segment name	SEG99	
$SEG99SIZ$	Audit security segment size	50 cylinders	

Customizing Communications

Data Object Broker – VTAM Application Definition and Nodename

VTAM is not supported between TIBCO Object Service Broker components—client processes, Execution Environments, Data Object Brokers, and external database gateways on a z/OS system.

VTAM is supported for VTAM terminals and the Execution Environments. Refer to member OSRUN in the distribution source library <HLQ>.FILEJCL for non-customized reference to the variables listed below. Modify OSRUN in the JCL data set to your new values and submit the job to check it out. For detailed instructions, see Appendix B, Configurations for Communications, Configuring VTAM Communications on page 411.

Variable	Description	Default	Your Value
TDS	Data Object Broker Communications Identifier	OSBDOB	
$NODNAM$	Data Object Broker name in a distributed data environment	DOB60ZOS	

Data Object Broker – TCP/IP Identifiers

If you are using TCP/IP to communicate between TIBCO Object Service Broker components-client processes, Execution Environments, Data Object Brokers, and external database gateways on a z/OS system and TIBCO Object Service Broker components on z/OS or non-z/OS platforms, TCP/IP identifiers will require customization. Refer to the RELAYCFG member in the distribution source library <HLQ>.FILECTL for non-customized references to the variables listed below. Modify member RELAYCFG in the CNTL data set to your new values and submit member OSRUN to check it out. For detailed instructions, see Appendix B, Configurations for Communications, Configuring TCP/IP on page 423.

Variable	Description	Default	Your Value
TDS	Data Object Broker communication identifier	OSBDOB	
$HOST$	Data Object Broker host name	ZOS.OSB.COM	
$DOBPORT$	Data Object Broker port number	9000	

Variable	Description	Default	Your Value
$EENAME$	Execution Environment communication identifier	OSBNEE	
$EEPORT$	Execution Environment port number	9001	

Customizing the Date Display Mask

Member USERMOD1 in the JCL data set contains JCL to modify the operands of the $SYSDATE macro in the MACRO data set. TIBCO Object Service Broker has a default date display mask of YYYY-MM-DD. If this format is acceptable, skip this step.

SMP/E conventions require that all components required by the installation (such as the TIBCO Object Service Broker base component, TIBCO Service Gateway for CICS, TIBCO Service Gateway for IMS TM, and so on) must be received and applied prior to applying USERMODs. If other components are to be installed, delay execution of this step until all other components have been applied.

Changing the Default Date Display Mask

STEP 1: **Customize USERMOD1 by modifying the $SYSDATE macro input.**

Example:

```
$SYSDATE DEFAULT='YYYY/MM/DD'
```

The mask you choose must have a length other than six (6). For valid alternate display masks, see to Valid Display Masks on page 38.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Changing the Default Date Display Mask After Applying USERMOD1

STEP 1: **Reapply the USERMOD specifying the REDO option.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Apply USERMOD1 to All Data Object Brokers

STEP 1: **Run USERMOD1 on each Data Object Broker you install.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Valid Display Masks

Valid display masks are comprised of the following components and separators:

Format Code	Description	Example
W	One- or two-digit week # (of year), with no leading 0.	1 or 25
WW	Two-digit week # (of year).	01
WWW	Abbreviated weekday.	Thur
WWWW	Full weekday.	Thursday
M	Numeric month, with no leading 0 (1 or 2 digits).	3 or 10
MM	Numeric month (2 digits).	02
MMM	Abbreviated month.	Mar
MMMM	Full month.	March
D	Day in month, with no leading 0 (1 or 2 digits).	5 or 14
DD	Day in month (2 digits).	02
DDD	Day in year (3 digits).	074
YY	Last two digits in a year (not recommended).	98
YYYY	Full year.	2007
QQ	Two-character quarter.	2Q
JD	Julian date.	04.074
CC	Two-digit century.	20

The separator characters can be any of the following: `/ \ ; : , . * - blank`

If the date format is not specified, the default is used. We recommend that you do not use JD or two-digit year-date combinations as installation default date formats. With these date formats, incorrect data can result from an assignment of a string field to a date field.

Customizing the Century Range

Member USERMOD1 in the JCL data set contains JCL to modify the operands of the $SYSDATE macro in the MACRO data set. If a field has a data type of date semantic and the corresponding input field is defined as a two-digit year, the century value for the date is derived from the YYCENTURYRANGE operand in the $SYSDATE macro. TIBCO Object Service Broker has a default century value of 1980. If this value is acceptable, you can skip this step.

SMP/E conventions require that all components required by the installation (such as the TIBCO Object Service Broker base component, TIBCO Service Gateway for CICS, TIBCO Service Gateway for IMS TM, and so on) must be received and applied prior to applying USERMODs. If other components are to be installed, delay execution of this step until all other components have been applied.

Valid YYCENTURYRANGE Values

The century range can be 0 or a value between 1900 and 2100. If the value is 0, TIBCO Object Service Broker uses the current century as returned by the TIM macro as the century value. For example, if the current date is 2000.01.01 or later, the century is 20.

If YYCENTURYRANGE is not 0 (for example, a valid value like 1940), and the input year is greater than or equal to the year portion of YYCENTURYRANGE, the century portion of YYCENTURYRANGE is used. If the input year is smaller than the year portion of YYCENTURYRANGE, the century portion of YYCENTURYRANGE+1 is used.

Changing the Default Century Range

STEP 1: **Customize USERMOD1 by modifying the $SYSDATE macro input.**

Example:

```
$SYSDATE YYCENTURYRANGE='1980'
```

The mask you choose must have a length other than six (6). For valid alternate display masks, see Valid Display Masks on page 38.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Changing the Default Century Range After Applying USERMOD1

STEP 1: **Reapply the USERMOD specifying the REDO option.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Apply USERMOD1 to All Data Object Brokers

STEP 1: **Run USERMOD1 on each Data Object Broker you install to avoid date inconsistent century ranges.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (EDIT member)`

STEP 2: **Run USERMOD1.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMOD1 (SUB member)`

It should end with RC=4. Return code 4 is expected because the linkedit warning messages that appear, IEW2651W, IEW2454W, and IEW2656W, are normal and can be ignored.

Configuring Unicode Processing

You can configure the conversion, collation, and case processing of Unicode data in TIBCO Object Service Broker. A set of configuration files is provided that allows you the following configuration choices:

- Accept the default configuration data that is part of the TIBCO Object Service Broker system.
- Use one or more of the supplied configuration files to override the default values.
- Modify one or more of the supplied configuration files to override the default values.
- Specify conversions between Unicode and External User Syntaxes using files from the following site:

  http://source.icu-project.org/repos/icu/data/trunk/charset/data/ucm/

These source configuration files are read by the UNIGEN utility, which produces assembler source code. The source code is then assembled and the resulting object code is processed by SMP/E to replace the original configuration data in the system. The original data corresponds to the IBM-037 code page. There are no External User Syntaxes defined by default.

Five types of configuration data are used for the following:

- Unicode to EBCDIC Mapping (UNITOEBC)
- EBCDIC to Unicode Mapping (EBCTOUNI)
- Unicode Case Mapping (UNICASE)
- Unicode Collation (UNICOLL)
- Unicode to/from External User Syntax Mapping (UNIXC01-UNIXC16)

File Formats

Each of the first 4 source configuration files consists of lines no longer than 80 characters:

- Data lines can include comments (which follow an asterisk) after the required fields. The formats of data lines for the four types of files are shown below. The names in parentheses are the names of the files used to configure the system.
- Comment lines include an asterisk followed by a comment or are entirely blank.

Unicode to EBCDIC Mapping (UNITOEBC)

Data mapping lines contain two significant fields (separated by white space):

- A hex value (0000 to FFFF) corresponding to a Unicode character.
- A hex value (00 to FF) representing the code point for the corresponding EBCDIC character.

Example:

```
* TIBCO Object Service Broker Unicode to EBCDIC conversion file
* Based on EBCDIC code page IBM-037.
0030  F0    *The character '0'
0031  F1    *The character '1'
```

A Unicode character can be mapped only once. You can map more than one Unicode character to the same EBCDIC character.

EBCDIC to Unicode Mapping (EBCTOUNI)

Data mapping lines contain two significant fields (separated by white space):

- A hex value (00 to FF) representing the code point for the corresponding EBCDIC character.
- A hex value (0000 to FFFF) corresponding to a Unicode character.

Example:

```
* TIBCO Object Service Broker EBCDIC to Unicode conversion file
* Based on EBCDIC code page IBM-037.
F0  0030    *The character '0'
F1  0031    *The character '1'
```

An EBCDIC character can be mapped only once.

Unicode Case Mapping (UNICASE)

Case mapping lines contain three significant fields (separated by white space):

- A hex value (0000 to FFFF) corresponding to a Unicode character.
- A case indicator: either `U` or `u` if the character is uppercase, or `L` or `l` if the character is lowercase.

- A hex value, representing the Unicode point for the same character with the opposite case.

Example:

```
* TIBCO Object Service Broker Unicode Case Mapping File
* Based on Unicode locale en_US.
0041  U  0061   * A
FF22  U  FF42   * B
```

Unicode Collation (UNICOLL)

Data lines contain a single significant field:

- A hex value (0000 to FFFF) representing a Unicode code point.

The data lines list the code points in order of their collation. The file must contain 65,536 unique data lines to specify all possible code points.

Unicode to/from External User Syntax Mapping (UNIXC01-UNIXC16)

The format is described at the following site:

http://icu.sourceforge.net/userguide/conversion-data.html

Data lines contain three significant fields (separated by white space):

- Hex value (<U0000> to <UFFFF>) representing a Unicode code point.
- A one- or two-byte hex value (\xhh or \xhh\xhh) representing the single-byte or double-byte character.
- A value (|0 or |1 or |2 or |3) indicating the fallback code to be used for this mapping. Only codes 0 and 1 are honored by TIBCO Object Service Broker.

The fifth source configuration file type is a ucm (UniCode Mapping) file which specifies a mapping between Unicode and a user-defined external syntax. You can have up to 16 files of this type to map up to 16 different external user syntaxes.

Sample Unicode Configuration Files

The following sample configuration files are shipped with TIBCO Object Service Broker. These names are the member names in the UNICODE data set provided. The 3- or 4-digit numbers in the filenames refer to the IBM-xxx EBCDIC code page they are based on. You can use the files as they are, or you can modify copies of these files to create the desired configuration specification.

Unicode to EBCDIC Mapping	EBCDIC to Unicode Mapping	Unicode Case Mapping	Unicode Collation
		UCASEDEF	UCOLDEF
UE037	EU037		UCOL037
UE273	EU273		UCOL273
UE277	EU277		UCOL277
UE278	EU278		UCOL278
UE280	EU280		UCOL280
UE282	EU282		UCOL282
UE284	EU284		UCOL284
UE285	EU285		UCOL285
UE297	EU297		UCOL297
UE500	EU500		UCOL500
UE1141	EU1141		UCOL1141
UE1142	EU1142		UCOL1142
UE1143	EU1143		UCOL1143
UE1144	EU1144		UCOL1144
UE1145	EU1145		UCOL1145
UE1146	EU1146		UCOL1146
UE1147	EU1147		UCOL1147
UE1148	EU1148		UCOL1148

Specifying Unicode Configuration (Optional)

STEP 1: **Customize UNIGEN.**

You can choose the configuration files (if any) to modify. The UNIGEN in-stream PROC that appears in the JCL takes three parameters:

- P – the type of configuration file to be processed:
  - P=1 Unicode to EBCDIC mapping.
  - P=2 EBCDIC to Unicode mapping.
  - P=3 Unicode Case Mapping.
  - P=4 Unicode Collation
- IN – the member name of the input configuration file. For correct performance, you should choose files that correspond to the NLS code page of your system.
- OUT – the member name of the output assembler source that the UNIGEN program generates. The suggested values for OUT are UNITOEBC, EBCTOUNI, UNICASE, and UNICOLL to correspond to the values of P, 1-4 respectively.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `UNIGEN (EDIT member)`

STEP 2: **Run UNIGEN.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `UNIGEN (SUB member)`

This will generate assembler source code in `<HLQNONV>.<SMP>.SRCSAMP`. It should end with RC=0.

STEP 3: **Modify USERMODA to match ++SRC statements with OUTPUT members generated in the previous step.**

Pay close attention to DD statements that are commented out as they may relate to TIBCO Service Gateways also installed in your site.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMODA (EDIT member)`

STEP 4: **Run USERMODA.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMODA (SUB member)`

It should end with RC=4. Return code 4 is expected because the binder warning messages that appear, IEW2609W, IEW2646W, and IEW2651W, are normal and can be ignored.

Specifying External User Syntaxes (Optional)

STEP 1: **Customize UNIGENXC.**

It invokes the UNIGEN utility to define mappings between Unicode and external user syntaxes. The UNIGENXC in-stream PROC takes four parameters:

- N – a 2-digit number of the external syntax (01 to 16)
- CP – the name of the codepage represented (such as IBM-939)
- IN – the member name of the input `ucm` file describing the mapping.
- FB – a TRUE or FALSE value indicating whether or not fallback codes present in the input file are to be used.

Customize UNIGENXC as shown above. You can run the PROC up to 16 times to define up to 16 external user syntaxes.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `UNIGENXC (EDIT member)`

STEP 2: **Run UNIGENXC.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `UNIGENXC (SUB member)`

This will generate assembler source code in `<HLQNONV>.<SMP>.SRCSAMP`. It should end with RC=0.

STEP 3: **Modify USERMODC to match ++SRC statements with OUTPUT members generated in the previous step.**

Pay close attention to DD statements that are commented out as they may relate to TIBCO Service Gateways also installed in your site.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMODC (EDIT member)`

STEP 4: **Run USERMODC.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMODC (SUB member)`

It should end with RC=4. Return code 4 is expected because the binder warning messages that appear, IEW2609W, IEW2646W, and IEW2651W, are normal and can be ignored.

Customizing the Message Prefix and SMF Subsystem Identifier

You can update and execute the SMP USERMODE member in the JCL data set if you wish to customize the three character message prefix or the four character SMF subsystem identifier.

Changing the Default Message Prefix

Messages in this release of TIBCO Object Service Broker begin with the prefix S6B (as in the message S6BKS010I- DATA OBJECT BROKER VERSION ...). Customers with extensive investment in automated operations may wish to change S6B to another prefix value, such as HRN, for compatibility with older releases of TIBCO Object Service Broker and ObjectStar. You can use any three-character prefix by updating USERMODE in your JCL data set and then applying SMP USERMODE. For example, to change the prefix from the default of S6B to HRN, set the PROD macro parameter ID= in USERMODE, as follows:

From: `PROD ID=S6B,SMFID=S6B`

To: `PROD ID=HRN,SMFID=S6B`

Applying USERMODE will then change the message prefix for all messages as specified.

Installations may have other procedures dependent on the value of the message prefix. Consult with your z/OS systems programmer or automated operations department before changing the message prefix.

Changing the SMF Subsystem Identifier

TIBCO Object Service Broker SMF records are identified by a unique four character SMF subsystem identifier. By default, this identifier is S6B but previous releases of the product used HRN. By customizing the SMP USERMODE member in the JCL data set, any four character value can used. This will cause all TIBCO Object Service Broker SMF records to have this value in the SMF SSI field. Refer to *TIBCO Object Service Broker for z/OS Monitoring Performance* for the layout and interpretation of SMF records. The SMF Subsystem Id field is at offset 18 (x'12') from the length field of the VBS record and is generally called XXXXSSI, where XXXX is a name dependent on the record subtype.

For example, to change the default SMF subsystem identifier from S6B to a four-character value of OSB, change the PROD macro parameter SMFID= in USERMODE as follows:

From: `PROD ID=S6B,SMFID=S6B`

To: `PROD ID=S6B,SMFID=OSB`

Applying USERMODE will then change the SMF subsystem identifier for all TIBCO Object Service Broker generated SMF data as specified. TIBCO Object Service Broker SMF utilities will automatically recognize SMF data using this new value.

Installations may have other procedures dependent on the value of the SMF subsystem identifier. Consult with your z/OS performance and accounting departments before changing this value.

Creating Execution Environment Parameter Modules

The following Execution Environment parameters are referenced in this procedure:

Parameter	Description	Default Value	Your New Value
MDL=MDL	EE communication identifier pattern	OSB9999	
SVC=$SVCNUM$	Type-3 TIBCO Object Service Broker SVC number	0	
TDS=TDS	Data Object Broker communication identifier	OSBDOB	

STEP 1: **Modify the Batch EE parameter module.**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `PARMBAT (EDIT member)`

STEP 2: **Modify the TSO EE parameter module.**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `PARMTSO (EDIT member)`

STEP 3: **Modify the Native EE parameter module (optional).**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `PARMNEE (EDIT member)`

STEP 4: **Modify the CICS EE parameter module (optional).**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `PARMCICS (EDIT member)`

STEP 5: **Modify the IMSTM EE parameter module (optional).**

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `PARMDC (EDIT member)`

STEP 6: **Uncomment EXEC statements that correspond to customized PARM*xxx* members from previous steps.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `EECONFIG (EDIT member)`

STEP 7: **Run EECONFIG.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `EECONFIG (SUB member)`

It should end with RC=0.

Installing the TIBCO Object Service Broker SVC

To run TIBCO Object Service Broker authorized and to use Cross Memory Services with other Execution Environments like CICS and IMS TM, you must install the TYPE(3) SVC provided with TIBCO Object Service Broker.

The following methods are used to install the TIBCO Object Service Broker SVC:

- Define a static SVC in an LPA eligible library using member IEASVCxx in SYS1.PARMLIB.
- Dynamically define and install the SVC using the S6BSVCIN TIBCO Object Service Broker utility.

TIBCO Object Service Broker must run authorized and the TIBCO Object Service Broker SVC is required under the following conditions:

- Password encryption is used.
- The FORCE CANCELUSER facility (HFUS transaction) is used.
- SMF recording for TIBCO Object Service Broker is required.
- VTAM SRB processing is desired.

Refer to Authorizing the Service Gateway for CICS on page 98 for more information.

Define the TIBCO Object Service Broker SVC Statically

STEP 1: **Assign a TYPE(3) SVC number for the use of TIBCO Object Service Broker.**

This will be the value of parameter SVC=$SVCNUM$ in Creating Execution Environment Parameter Modules on page 50. See *TIBCO Object Service Broker Parameters* for details.

STEP 2: **Define the SVC in member IEASVCxx in SYS1.PARMLIB.**

Sample definition:

```
SVCPARM nnn,REPLACE,EPNAME(S6BSVC)TYPE(3),
APF(NO),NPRMPT(NO)
```

nnn represents the number of the SVC assigned by your site:

Member in: `SYS1.PARMLIB`

Member: `IEASVCxx`

STEP 3: **Place the SVC module S6BSVC in an LPA eligible library.**

Member in: `<HLQNONV>.<INSTVER>.AUTH`

Member: `S6BSVC (COPY member)`

STEP 4: IPL the CPU to activate the SVC**.**

Define the TIBCO Object Service Broker SVC Dynamically

STEP 1: **Ensure that the SVC number you will use is defined to the image of z/OS where you will run your Execution Environment (confirm this with your site's systems programmer).**

Refer to *TIBCO Object Service Broker Parameters* for a description of the SVC parameter.

STEP 2: **Run SVCINSTL.**

Note the following:

- This JCL can be modified and run as a started task if desired.
- Program S6BSVCIN can be executed during an IPL using member IEACMDxx in SYS1.PARMLIB, if the SVC is to be dynamically installed.
- If you need to replace the SVC at any time, code PARM='S6BDRCB0,REPLACE' in the EXEC statement. The existing installed TIBCO Object Service Broker SVC is then replaced in the system. Example:

  ```
  EXEC PGM=S6BSVCIN,PARM='S6BDRCB0,REPLACE'
  ```

- Execute program S6BSVCIN from an authorized library.
- Do not copy the SVC to a system library such as SYS1.LPALIB. The SVC must be in the SVCINSTL's STEPLIB DD concatenation.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `SVCINSTL (SUB member)`

To delete an old release level of the SVC and install a new level, delete (DELETE) the SVC with the load library of the old release and install (ADD) the new version with the new load library. Doing so eliminates any mismatches that might occur during authentication of the S6BSVCIN utility and processing of the SVC.

Customizing the Operating Environment

The Data Object Broker can be executed from either a batch job or a started task. You can do further customizations to its operating environment.

Step1: Customize the Data Object Broker Initialization Parameters

Data Object Broker initialization parameters are read in at startup. Review each parameter and ensure its value is appropriate for your site. Most parameters should be acceptable as provided.

For information on Data Object Broker parameters, refer to *TIBCO Object Service Broker Parameters*.

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `OSRUNPRM (EDIT member)`

Step 2: Customize the Data Object Broker Startup JCL and EXEC

STEP 1: **Use OSRUN to start the Data Object Broker as a batch job. It is identical to S6A8DOB in <HLQ>.<INSTVER>.OSB.JOBS. Customize further as required.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `OSRUN (EDIT member)`

STEP 2: **Use OSRUNSTC to start the Data Object Broker as a started task. Customize further as required and move it to a system procedure library.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `OSRUNSTC (EDIT member)`

Step 3: Customize TSO Execution Environment EXEC

STEP 1: **Use USER to log in as a typical TIBCO Object Service Broker user or developer. Customize further as required.**

Member in: `<HLQNONV>.<INSTVER>.CLIST`

Member: `USER (EDIT member)`

STEP 2: **Use ADMIN to perform operational functions. Customize further as required.**

Member in: `<HLQNONV>.<INSTVER>.CLIST`

Member: `ADMIN (EDIT member)`

Step 4: Customize the Data Object Broker Spin Processing JCL

During normal Data Object Broker operations, journal data sets become full and require copying. The process of varying a full journal offline, copying its contents, and resetting its status is referred to as a *spin*. Journals contain modified data pages that have been checkpointed. The spin process copies pages from the journal to a DASD generation data group. Members SPIN01 and SPIN02 in the JCL data set are used to perform this task. SPIN01 dumps journal 1 and SPIN02 dumps journal 2. Before starting the Data Object Broker, you must customize the JCL that you plan to use to spin the journals. How you modify the JCL is determined by whether you run the spins as batch jobs or as started tasks.

Run SPIN01 and SPIN02 as Batch Jobs

STEP 1: **Review the JCL and customize further as required.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `SPIN01 (EDIT member)`

Member: `SPIN01 (EDIT member)`

STEP 2: **Make sure that OSRUNPRM specifies the following:**

- SPINOPTION=JOB
- SPINDSNAME=<HLQNONV>.<INSTVER>.JCL (contains SPIN01 & SPIN02)
- SPINMEMBER=SPIN0

The SPINxx jobs are verified in Chapter 6, Installing the Service Gateway for IMS TM, on page 121.

If SPINxx JCL requires modification, you can shut down and restart the Data Object Broker to make the changes take effect, or you can issue the following Data Object Broker operator command:

```
SPINLOADJCL=nn.
```

This refreshes the JCL in memory for the specified journal number without requiring that the Data Object Broker be restarted. The SPINLOADJCL command can be truncated to SPINLOAD. For more information on Data Object Broker operator commands, refer to Commands to Control a Data Object Broker on page 260.

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `OSRUNPRM (EDIT member)`

Run the SPIN as a Started Task

STEP 1: **Review the PROC and customize further as required. Move it to a system procedure library.**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `SPINSTC (EDIT member)`

STEP 2: **Make sure that OSRUNPRM specifies the following:**

- SPINOPTION=STC
- SPINMEMBER=SPINSTC

If SPINSTC JCL is modified, JCL takes effect the next time a START SPINSTC command is issued

Member in: `<HLQNONV>.<INSTVER>.CNTL`

Member: `OSRUNPRM (EDIT member)`

Step 5: Customize the Data Object Broker Spin Merge JCL

The SPINMRG job is automatically submitted by the SPINxx jobs when the GDG limit of the journal spinouts is reached. This limit is set by OSEMOD variable $SPINLIM$, which has a default value of 2. A spinout GDG is created whenever a journal fills up and the corresponding SPINxx job is submitted. For example, with a $SPINLIM$ default of 2, SPIN01 runs when journal 1 fills up then SPIN02 runs when journal 2 fills up. Job SPIN02 will then release SPINMRG.

Review the JCL and customize further as required

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `SPINMRG (EDIT member)`

Step 6: Customize the Data Object Broker Continuous Backup JCL

BKUPCON is released by a SPINMRG job. It is verified in Chapter 3, Performing Postinstallation Tasks, page 61.

Review the JCL and customize further as required

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `BKUPCON (EDIT member)`

Step 7: Query the Capabilities of Your CPU

The S6BBRHDW member in the JCL data set contains JCL to run the S6BBRHDW (CPU Capabilities Report) utility, to report on the availability of the hardware instructions required to run TIBCO Object Service Broker for z/OS.

Produce the CPU Capabilities Report. It should end with RC=0. If the report shows that your system is missing instructions, then it will have some restrictions in running this version of the software; contact TIBCO Support to resolve the issue.

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `S6BBRHDW (SUB member)`

Chapter 3 Performing Postinstallation Tasks

This chapter describes how to perform the postinstallation tasks, including one-time tasks and ongoing ones, for the TIBCO Object Service Broker base component.

Topics

Overview

To ensure that your TIBCO Object Service Broker system runs optimally and to avoid issues, you must establish tasks at installation time and run certain monitoring tasks daily, weekly, bimonthly, or monthly. This chapter describes how to set up the required tasks once you have installed your base TIBCO Object Service Broker system.

Postinstallation Tasks

Perform the following tasks once you have completed the base installation to prepare your TIBCO Object Service Broker facilities:

- Run spin verification and perform cleanup tasks.
- Customize the `@SCHEDULEMODEL` table.
- Schedule purges of the `AUDITLOG` table.

See the sections that follow for details.

In addition, you must ensure the following:

- The installation is customized for your environment, in particular that the Execution Environment and Data Object Broker parameters are set to the correct values for your types and volumes of work. See Chapter 2, Customizing the Installation.
- Communications are functioning. See Appendix B, Configurations for Communications.

Scheduled Tasks

Once your installation is complete, schedule tasks to monitor your TIBCO Object Service Broker environment. The nature of the tasks determines whether you should perform them daily, weekly, bimonthly, or monthly. Here are a few questions to answer:

- What is the status of the previous night's backup?
- Are there application updates, including those for new applications?
- What are the performance statistics?
- Are there software updates?
- Is the spacing adequate for growth?

For details, see Scheduling of Monitoring Tasks on page 76.

Spin Verification and Cleanup

Spin verification takes you through a forced journal spin to familiarize you with the TIBCO Object Service Broker journal-processing and merging process. When running TIBCO Object Service Broker in production, the Data Object Broker spins the journals automatically.

Prerequisite Tasks

Before verifying spin processing, be sure to backup the system in full by running member BACKUP as part of the initial installation. If you did not perform this task before, do it now. Spin verification requires a full system backup for the BKUPCON job.

Following are two prerequisite tasks.

Installation of the Initial TIBCO Object Service Broker Batch Server

A batch server facility in TIBCO Object Service Broker manages job queues for TIBCO Object Service Broker batch jobs that are initiated by the SCHEDULE statement. Install the initial batch server by following the procedure described in this section.

For details on operating and managing batch servers and queues, refer to Chapter 14, Managing Batch Processing, on page 241.

Step 1: Customize the @BATCH_JCL Table

Do the following:

1. Log in to TIBCO Object Service Broker as SYSADMIN with USER EXEC.
2. From the TIBCO Object Service Broker command line, invoke the TED tool and edit the table @BATCH_JCL and instance @DEFAULT, HURON by typing the following on the command line and pressing Enter:

   ```
   ex TED('@BATCH_JCL(@DEFAULT,HURON)')
   ```

   For table instance @BATCH_JCL(@DEFAULT,HURON), change all DD statement data-set names from HURON.LOAD to *HLQNONV*.*INSTVER*.AUTH, where *HLQNONV* and *INSTVER* are OSEMOD installation variables (for example, OSB.R60.TST.AUTH). The data set-name must be in uppercase. For information on the TED tool, see the *TIBCO Object Service Broker Shareable Tools* manual.

3. From the TIBCO Object Service Broker command-line, invoke the TED tool and edit the table @BATCH_JCL and instance @DEFAULT, JOBCARD by typing the following on the command line and pressing Enter:

   ```
   ex TED('@BATCH_JCL(@DEFAULT,JOBCARD)')
   ```

   For table instance @BATCH_JCL (@DEFAULT,JOBCARD), change the JOB card ACCOUNT# information to the appropriate values for your site.

   When updating the JCL images in the @BATCH_JCL table, ensure that the individual JCL statements are shorter than 71 bytes. All standard JCL rules apply. After saving the JCL, a warning message is displayed, stating that data is to be truncated. The truncated data is in columns 73 to 76, which are not used for coding JCL. This message is expected and you can ignore it.

4. Leave the following substitution variables as is:

{MDL}	Replaced with the Execution Environment communications identifier model.
{QUEUE}	Replaced with the queue name.
{TDS}	Replaced with the Data Object Broker communications identifier with which the batch server is to communicate.
{USERID}	Replaced with the requestor's user ID as specified in the REQUESTOR ID field of the Submit Batch Request screen.

5. Press PF3 to save the changes.

Step 2: Define an Initial Batch Queue

Do the following:

1. Log in to TIBCO Object Service Broker as SYSADMIN with USER EXEC.
2. Execute the BATCH tool from the EX execute rule menu option or from the command line and then press Enter, as follows:

   ```
   COMMAND  ==>     EX  BATCH
   ```

   The BATCH submission menu is displayed. For details on the BATCH tool, see the *TIBCO Object Service Broker Shareable Tools* manual.

3. Position your cursor beside the option Queue Definition and press Enter.

   The Batch Submission Facility Queue Definition screen is displayed.

4. Type one of the following values and press Enter:
   - Your initial queue name, for example, SITEQUEUE
   - Your site value for the OSEMOD installation variable *$BATQNM$*.
   - ADMIN, which is the default queue definition
5. On the next screen, Queue Definition, type your site-selected parameters in at least the following two fields and then press PF3 to save your changes.
   - **Wait Duration:** The time in seconds that the batch server waits (for example, 3600) if the queue becomes empty.
   - **Wait Limit:** The number of times the batch server goes into a wait state before shutting down; for example, 8.
6. Press PF3 again to save and exit.

Step 3: Prepare the Initial Batch Server

Sample JCL in member BATSRVL1 in JCL invokes the batch server.

OSEMOD variable *$BATSRI$*, which defaults to XBATCHL1, defines the member name in the CNTL data set for the default batch server's initialization parameters. Customize members BATSRVL1 and XBATCHL1 with the OSEMOD ISPF edit macro and then edit the startup parameters. See the following subsections.

Start-up Parameters

Member XBATCHL1 in CNTL contains sample startup parameters. Note these two rules:

- Specify each parameter on a separate line.
- Code parameter statements as PARM=VALUE with no spaces on either side of the equal (=) sign.

Here are the parameters:

MDL=	The pattern for the Execution Environment communications identifier adopted by a TIBCO Object Service Broker server for communications. If not specified, the values defaults to *TDS*.
QUEUE=	The name of the initial queue you defined with the Batch Submission facility in a previous step. The OSEMOD default is ADMIN.
TDS=	The Data Object Broker communications identifier.

USERID=	The Data Object Broker login ID for the batch server. The OSEMOD installation default is BATSRVL1.

Batch Server

You can run the batch server as a batch job or started task. For the latter, be sure to first complete the z/OS security setup. The batch server submits jobs queued to it with an internal reader. Place BATSRVL1 in your system PROCLIB, that is, SYS1.PROCLIB.

Here are the related references:

- For information on operating the batch server, see Chapter 14, Managing Batch Processing, on page 241.
- For information on running batch jobs, see the *TIBCO Object Service Broker Programming in Rules* manual.

For more details on running TIBCO Object Service Broker batch applications, see the *TIBCO Object Service Broker for z/OS External Environments* manual.pf

Creation of Education Workshop Objects

Create education workshop objects as follows:

Step 1: Log In to TIBCO Object Service Broker

To create a workshop environment, do the following:

1. Start the Data Object Broker.

   For details, see Step 3: Customize TSO Execution Environment EXEC on page 56.

2. Log in to TIBCO Object Service Broker with USER EXEC (with user SYSADMIN, whose default password is SYSADMIN). For example:

   ```
   TSO EX 'HLQNONV.INSTVER.CLIST(USER)' 'U(SYSADMIN) P(SYSADMIN)'
   ```

Step 2: Create User IDs

Create TIBCO Object Service Broker user IDs in the Security interface. For details on the procedure, see the TIBCO Object Service Broker Managing Security manual.

Step 3: Create Table Instances

From the administrator workbench, execute the SETUP_EDUC(*userid*) rule, which creates instances in two parameterized tables: @EMPLOYEE and @DEPARTMENT. The argument *userid* is the parameter value for each table instance. For details on creating tables, see the TIBCO Object Service Broker Managing Data manual.

Do the following:

1. Execute once for each student to create his or her table instances.
2. Execute the rule with the instance value EDUC.

Assumptions

Table additions and updates are recorded as journal data after the Data Object Broker has performed a checkpoint. For the purpose of this activity, it is assumed that you must spin your journals and that you are using the continuous backup process. With that process, you merge spun journal data with the current backup to maintain a current system backup.

Spin Verification

TIBCO Object Service Broker merges journal accumulation data after every nth journal spin, where n is set to 2 during the initial installation by the OSEMOD variable *$SPINLIM$*. That value determines the number of times members SPIN01 and SPIN02 are submitted before member SPINMRG merges journal data into a single data set.

If you create the spins as jobs, one member exists for each journal up to a limit of 255. However, if you create them as started tasks, only one procedure applies, with the journal defined by JRNLDSN.

To verify the spin, follow these steps:

Step 1: Spin the Active Journal

When the Data Object Broker is running, it always contains an active journal data set. The spin process is caused by the data set becoming full or by an operator request, called a forced spin.

To force a journal spin, run this z/OS operator command from a z/OS operator console:

MODIFY *jobname*,SPINSUBMIT=I

where *jobname* is the name of the Data Object Broker for which a journal spin is forced.

Next, the Data Object Broker acknowledges the spin request. TIBCO Object Service Broker then starts a *SPINxx* job to copy the data from the active journal to an archive data set (SPINOUT). For your initial verification of the installation spin, this data contains page images produced during any postinstallation tasks, such as creation of the education workshop tables, batch server installation, and so on.

Reexecute the prerequisite tasks as described on page 63 to write out data to the other journal. Afterwards, reenter the z/OS operator command to invoke, submit, and run the other *SPINxx* job.

Spin jobs result in the following:

- The first step in both SPIN01 and SPIN02 produces a return code of 0.
- The second step in both jobs produces a return code of 0.
- The third step in each job, TESTSPN, returns the number of journal generation data groups (GDGs) in existence.
- Since the spin limit is 2 and you performed two journal spins, member SPINMRG is submitted by step SPAWN. SPINMRG merges journal accumulation GDGs into one manageable output data set and produces a return code of 0.

Step 2: Test the Continuous Backup JCL

The BKUPCON member in the JCL data set is the continuous backup job. It sorts the page images, merges them with the journal accumulation data, and then combines them with the latest backup to create a current backup.

To test the BKUPCON JCL, submit the job and ensure that it ends with RC=0.

Now that you have verified the journal spin, consider resetting the *$SPINLIM$* value to something more appropriate for your site with the following steps:

1. Delete and redefine the GDG base with the ENTRIES parameter, specifying the number of generations required.
2. Update the first condition test on step SPAWN of SPIN01 and SPIN02.

   With *$SPINLIM$* set to 2 initially, the condition test is coded in the JCL as (2,GT,TESTSPN). Update the number 2 to the new spin limit. The maximum number of journal spin GDG data sets is defined by the OSEMOD variable *$JSRGDG$* in STEP6 in member S6A3ALOC of the OSB.JOBS data set.

Cleanup

This section describes the cleanup process.

Backup Procedures

Create a procedure that suits your site requirements for merging the journal data into the current backup. For details on planning production backup and recovery procedures, see the *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* manual.

The backup procedures are critical to the availability of your TIBCO Object Service Broker system. Involve the system administrator, operations staff, and systems programmer, and other appropriate team members when creating and documenting those procedures. Afterwards, test them.

Backup Processing

The `BACKUP` member in the JCL data set serves as sample JCL to back up all page data sets within a segment. It produces a copy of the TIBCO Object Service Broker page data sets that you can restore with the `S6BTLRPS` utility. You also use `S6BTLRPS` before relocating page data sets. For details, see the *TIBCO Object Service Broker for z/OS Utilities* manual.

Member `BACKUP` uses the GDG created during installation for its output data set. That data group is defined in `STEP6` of member `S6A3ALOC` in the `OSB.JOBS` data set.

You cannot back up or relocate page data sets with `IDCAMS REPRO` because it would destroy the structure of the internal page data set.

Dump Processing

Members `OSRUNSTC` and `OSRUN` in Data Object Broker JCL contain a `SYSMDUMP DD` statement. That data set is part of a GDG defined in `STEP6` of member `S6A3ALOC` in the `OSB.JOBS` data set with a limit of `5`, which is defined by the `OSEMOD` variable *`$DMPGDG$`*.

Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table

During installation, you must customize some of the supplied @SCHEDULEMODEL table instances. In particular, tailor the JCL statements in several table instances to reflect your site's standards and naming conventions. You can execute in batch mode through the @SCHEDULEMODEL table by means of the SCHEDULE statement. @SCHEDULEMODEL JCL is also used in the Promotion system.

Here are the related references:

- For information on the @SCHEDULEMODEL table and the SCHEDULE statement, see the *TIBCO Object Service Broker Programming in Rules* and *TIBCO Object Service Broker Application Administration* manuals.
- For details on @SCHEDULEMODEL JCL for promotions, see the *TIBCO Object Service Broker Managing Deployment* manual.

Step 1: Log In and Invoke the Table Editor

Do the following:

1. Log in to TIBCO Object Service Broker as SYSADMIN with USER EXEC.
2. Invoke the Table Editor for each of the specified instances. For example, on the command line, type the following and press Enter:

```
COMMAND  ==>    ED  @SCHEDULEMODEL('MVS','*DEFAULT*')
```

Step 2: Edit Table Instances

Edit the instances in table @SCHEDULEMODEL, as follows:

1. For all table instances, customize the JCL JOB card parameters: account code, name, and message class.

   Table instance @SCHEDULEMODEL (MVS,*DEFAULT*) is for general users.

2. For all table instances, except @SCHEDULEMODEL (MVS, PROM_ALLOC), change all DD statements with a last qualifier of AUTH to *$HLQNONV$*.*$INSTVER$*.AUTH, where *$HLQNONV$* and *$INSTVER$* are OSEMOD installation variables; for example, OSB.SMP60.AUTH.

Pay particular attention to the JOB card for jobs submitted by multiuser address spaces, such as the Native Execution or CICS environments. Unless your z/OS external security product has cross-authorized the environmental user identification to submit jobs on behalf of TIBCO Object Service Broker-defined users, your jobs might fail on submission if you have not deleted or commented out the parameter USER={USER} on the associated @SCHEDULEMODEL job card.

When a rule causes a job submission, TIBCO Object Service Broker automatically substitutes the parameter within the curly brackets ({}) with the current value for the parameter within the brackets. For user identifications such as SYSADMIN, the external security manager is unlikely to recognize the user identification SYSADMIN on the JOB card and, subsequently, Job Entry Subsystem (JES) fails the job before execution with a message like this one:

```
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
```

or this one:

```
IEF642I EXCESSIVE PARAMETER LENGTH IN THE NOTIFY FIELD.
```

If the NOTIFY parameter is used, the job could fail with the following message:

```
IKJ144I UNDEFINED USERID(S) uuuuuuu
```

Table Instances and Functions of JCL

The following table describes JCL's table instances and functions.

MVS,*DEFAULT*	Applies if the object of a SCHEDULE command is a rule and no matching @SCHEDULEMODEL instance exists.
MVS,@HLL_COBJCL	Processes TIBCO Object Service Broker statements embedded in a COBOL program. It sends the source to the HLI preprocessor in the TIBCO Object Service Broker environment and then redirects the output as input to a PROC to compile and link COBOL. To learn how to use COBOL programs with the TIBCO Object Service Broker host language interface, see the *TIBCO Object Service Broker for z/OS External Environments* manual.
MVS,APPLY_CHANGE	Used by the Promotion system to apply a change request in a source system in batch.
MVS,APPLY_PROM	Used by the Promotion system to apply a change request in a target system in batch.
MVS,PROM_ALLOC	Allocates data sets used by the Promotion system.

MVS,SPOOLSTRIP	Prints output in multiple session environments, such as CICS or Native Execution. Ensure that the External Environment parameter PRINTSPOOLCLASS points to a non-HELD JES SYSOUT output class. Otherwise, the following error occurs: S6BDP001E - IEFSSREQ ERROR - R15=X'00' - SSOBRETN=X'04'
MVS,TREE	Used by the PR print rules option on the workbench if Hardcopy is set to Y on a print-tree request.
MVS,XRF	Used by the PR print rules option on the workbench if Hardcopy is set to Y on a cross-reference request.

Specified Variables

Do *not* change the following substitution variables, which represent Execution Environment parameters. For details on those parameters, see the *TIBCO Object Service Broker Parameters* manual.

{ACTION}	The type of invocation for the initial user-rule specified in the RULE parameter.
{BROWSE}	A specification of whether the session's first transaction can update TDS tables and external databases.
{CHARSET}	The default national character set.
{CLASS}	The default JES SYSOUT class for output generated with the TIBCO Object Service Broker print facilities. **Note:** In a JES3 environment, ensure that the output class specified in the user profile options is one that is defined to JES3 as External Writer Held. The SYSUT2 statement in the SPOOLSTRIP JCL contains a hard-coded class so that no substitution of the Class parameter from the user profile occurs.
{COPIES}	The number of copies to print of a report generated by TIBCO Object Service Broker.
{DB2LOG}	A specification of whether the Execution Environment is to log DB2 server commands for the purpose of creating STATIC SQL handlers.
{DEST}	The default printer destination for output generated by TIBCO Object Service Broker.
{FCB}	The name of a Forms Control Buffer (FCB) to be used during the printing of output generated by TIBCO Object Service Broker.

{FORM}	The name of a form on which JES SYSOUT output generated by TIBCO Object Service Broker is to be printed.
{INSTLIB}	The name of the library that holds the rules for the installation promoted from the local libraries.
{LIBRARY}	The name of the local library for rules calls.
{MDL}	The model Execution Environment communications identifier.
{PARM}	The arguments for the rule for batch processing. Wraps alphanumeric values in double single quotation marks: two at the start and two at the end.
{PARS}	The arguments for the rule for batch processing. Wraps alphanumeric values in single single quotation marks: one at the start and one at the end.
{PASSWORD}	The user's login password to be passed to TIBCO Object Service Broker.
{PRINTDEST}	The default printer destination for output generated by TIBCO Object Service Broker.
{RULE}	The name and arguments of the first application rule to be invoked after processing of login rules. Does not apply to SDK (C/C++) or SDK (Java) sessions.
{SEARCH}	The library search environment for the first rule to be executed.
{SYSTEMLIB}	The name of the rules library that contains the rules shipped with TIBCO Object Service Broker.
{TDS}	The communications identifier (COMMID) of the Data Object Broker to which to connect.
{TEST}	A specification of whether to run user sessions in test mode.
{UCS}	The name of a universal character set (UCS) for printing the JES SYSOUT output generated by TIBCO Object Service Broker.
{USERID}	The session user ID for TIBCO Object Service Broker.
{VARLDELIMITER}	The character that serves as the left delimiter for enclosing substituted variables in @SCHEDULEMODEL.
{VARRDELIMITER}	The character that serves as the right delimiter for enclosing substituted variables in @SCHEDULEMODEL.

{XWTR}	The name of an external writer (XWTR) for printing output generated by TIBCO Object Service Broker.

For details on the PR print rules option on the workbench, see the section on the RULEPRINTER tool in the *TIBCO Object Service Broker Shareable Tools* manual.

Do *not* name a print destination in the JES complex S6BDRPRT, which is the name of the TIBCO Object Service Broker offline print utility that generates and prints output from the Native Execution Environment and CICS. S6BDRPRT runs the SPOOLSTRIP job, which operates incorrectly if the print destination is called S6BDRPRT.

Step 3: Save Changes

Press PF3 to save the changes.

Security Framework for Archiving the Audit Log

Be sure to purge the data in the Audit Log. Otherwise, the segment that holds audit data in the ACCESSLOG table (usually segment 99) eventually fills up, terminating processing for TIBCO Object Service Broker. Such an occurrence might require a recycling of your Execution Environment.

To learn how to set up the security framework for archiving the ACCESSLOG table, see the chapter Archiving the Audit Log Data in the *TIBCO Object Service Broker Managing Security* manual.

Scheduling of Monitoring Tasks

For information on scheduling monitoring tasks, see the following references:

Facility	Reference
Administration Menu	Chapter 18, Monitoring and Controlling the TIBCO Object Service Broker Environment
HMON	*TIBCO Object Service Broker for z/OS Performance Monitoring* manual
Mainframe Service Tracker	Chapter 17, Using the Interface to TIBCO Mainframe Service Tracker

The frequency of the tasks is determined by your operational requirements. The following are general recommendations only.

Daily Tasks

Schedule the following daily tasks:

- Check the status of the previous night's backup.
- Browse the Data Object Broker JES log for messages that warn of segment space problems, for example:

  ```
  S6BKP058A- WARNING, SEGMENT=segment.name IS  nn% FULL.
  ```

  To view the percentage of free pages in each segment, choose the `ADMIN` utility option `B SEGMENT/DASD`.
- Browse the Data Object Broker JES log for messages that warn of issues in resource definitions, for example:

  ```
  S6BKC036L iiiiiiii REJECTED TYPE=PEER DOB REASON=return_code /
  reason_code
  ```
- Browse all multiuser Execution Environment JES logs for any unusual error messages or rule failures. Follow-up on the issues, if any.

Weekly Tasks

Schedule the following weekly tasks:

- Check for application updates, including those for new applications. Install the updates that are appropriate.

- Examine the performance statistics.
- If you have not set up continuous backup, perform offline segment backups, assuming that a window exists, and ensure that Batch Pointer Check (S6BBRPTR) shows that the backups are clean. If the time window is not sufficient for backups, run a DASD—not a backup—Batch Pointer Check against each offline segment. That task requires no backup but still confirms the physical consistency of the segments.
- If you are using peer processing or external database gateways, choose the ADMIN option 1 IN-DOUBT TRANSACTIONS to confirm that no in-doubt transactions are pending. In case of any, find out why TIBCO Object Service Broker has not automatically resolved them or resolve them manually.

Bimonthly or Monthly Tasks

Schedule the following bimonthly or monthly tasks:

- Verify that the test and certification systems are synchronized with the systems to which software updates are to be applied.
- Check for software updates and schedule maintenance at the most opportune time.
- Examine all multiuser Execution Environment JES logs for information on the amount of virtual storage space and create an ongoing profile of space availability. Review the message like the one below and record the available storage space:

  ```
  S6BST301I STORAGE AVAILABLE AT STARTUP IS  nnnnK OF 24 BIT AND
  nnnnnnK OF 31 BIT
  ```

- Select option A of the ADMIN utility to review the logical GET to physical READ processing. Strive for a successful hit ratio of at least 95 percent. See this formula:

  ```
  Hit_ratio =  [ (logical_GETs – physical-READs) * 100 ]  /
  logical_GETs
  ```

  If the hit ratio is substantially less than 95 percent, increase the resident page pool size (Data Object Broker parameter RESIDENTPAGES) to hold more page images in memory and avoid I/O to the segment page data sets.

You can manually create an ongoing processing profile of virtual storage space, DASD space, I/O, response times, and other metrics with such tools as ADMIN and HMON. However, a better approach is to activate the product-based SMF recording and automate the process of extracting, archiving, and analyzing the data.

Chapter 4 Initializing and Testing the Native Execution Environment

This chapter describes how to initialize and test the Native Execution Environment.

Topics

Overview

The Native Execution Environment is a TIBCO Object Service Broker component that enables multiple users concurrent online access to a Data Object Broker and its attached external database servers. It can be executed as a batch job or as a started task.

The single most important benefit of the Native Execution Environment is reduced overhead, thus accommodating a large number of user sessions in a single address space.

Facilities That Require a Native Execution Environment

The Native Execution Environment component is required for the following facilities:

- TIBCO Object Service Broker Eclipse UI.

  NOTE: The Service Gateway for CICS component can be used as an alternative to connect Eclipse UI sessions.

- A login to TIBCO Object Service Broker directly from VTAM
- IMS TM support
- Distributed data
- Service Gateway for Adabas, Service Gateway for Datacom, and Service Gateway for IMS/DB

Events During Native Execution Environment Initialization

During Native Execution Environment initialization, the following events take place:

- The Execution Environment establishes environments for Cross Memory Services (XMS), VTAM, and TCP/IP communications. For VTAM, it opens an ACB with the name specified by the EENAME initialization parameter.
- An Execution Environment is created that establishes communication with the Data Object Broker specified by the TDS parameter defined in the Execution Environment initialization parameters.
- Communications from the user to the Native Execution Environment and from the Native Execution Environment to the Data Object Broker are established using the most efficient communication method possible (Cross Memory Services or TPC/IP).

Logging In to the Native Execution Environment

When this initialization is complete, users can log in to the Native Execution Environment using the VTAM LOGON command. For details, see Log In to a Native Execution Environment on page 85.

IMS TM users connect to a Native Execution Environment when they invoke the TIBCO Object Service Broker IMS TM command to log in to TIBCO Object Service Broker.

See Also

TIBCO Object Service Broker for z/OS External Environments for more information about the Native Execution Environment.

TIBCO Object Service Broker Parameters for more information about the Execution Environment parameters.

TIBCO Object Service Broker Messages With Identifiers for messages produced by the Native Execution Environment.

Initialization Activities

Customize PARMNEE

Sample Native Execution Environment initialization parameters are provided in member PARMNEE of the CNTL data set. Review all the parameters contained in the member and specify proper values for each one.

Step NATIVE in the EECONFIG member in the JCL data set must be uncommented and run to create the default option module before starting a Native Execution Environment. For a description of the sample initialization parameters included in this step, refer to *TIBCO Object Service Broker Parameters*.

Hierarchy of Initialization Parameters

Initialization parameters can be specified in several ways:

1. Default environment parameters obtained from the environment option module
2. Additional parameters specified by the HRNIN DD statement
3. Parameters specified by coding the PARM= JCL keyword on the EXEC statement.

   All operating system restrictions apply when using this keyword; for example, the coded input string cannot be longer than 100 characters.

   Here is an example of how to code the PARM= keyword on the EXEC statement:

```
//STEP1 EXEC
         PGM=HRNDR000,PARM='TDS=TESTSRV,EENAME=HRNSRVB,CLASS=G'
```

   In this example, the Native Execution Environment establishes communications with a Data Object Broker that has a communications identifier of TESTSRV. The name of the Execution Environment VTAM ACB to be opened is HRNSRVB. Output generated by users while logged in is directed to SYSOUT class G.

This login example assumes that the necessary VTAM resources specified in the login command are correctly defined to the system prior to the login command being issued.

Define a VTAM ACB Definition for Each Native Execution Environment

The Native Execution Environment requires a dedicated VTAM application definition. Define the definition if it does not already exist. The application definition to be used by the Execution Environment is specified by the EENAME Execution Environment parameter. Refer to Define a VTAM ACB Definition for Each Native Execution Environment on page 415 for specific details.

Enable Use of the TIBCO Object Service Broker UI

Use the following steps to activate the connection to the TIBCO Object Service Broker UI:

1. Ensure that National Language Support (NLS) is turned on in your TIBCO Object Service Broker system. For information about how to turn on NLS in TIBCO Object Service Broker, refer to *TIBCO Object Service Broker National Language Support*.
2. Ensure that a TCP/IP port is assigned to the Execution Environment. For information about how to assign a TCP/IP port, refer to Configuring TCP/IP on page 423.
3. Modify the Execution Environment JCL to add the following:
   - $HLQNONV$.$INSTVER$.AUTH to the HRNEXTR concatenation.
   - STANDBYNUM=*nn* to the Execution Environment's HRNIN parameter file, where *nn* is a number greater than the number of concurrent TIBCO Object Service Broker UI sessions you expect on the Execution Environment. This parameter makes *nn* sessions available for TIBCO Object Service Broker UI users. For more information on this parameter, refer to *TIBCO Object Service Broker Parameters*.

For instructions on installing the TIBCO Object Service Broker UI, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Testing the Native Execution Environment

Start a Native Execution Environment

Overriding Default Parameters

To override any default parameters defined in the Native Execution Environment option module, specify them by the HRNIN DD statement or on the EXEC statement via the JCL keyword PARM= of member NATIVEEE (batch job) in the JCL data set.

Executing the Native Execution Environment

The Native Execution Environment can be executed as a batch job or a started task. It must execute from an authorized data set. For more information on making the load library authorized, see Configuring Authorized Libraries on page 22.

Starting the Native Execution Environment as a Batch Job

Member NATIVEEE in the JCL data set can be used to start the Native Execution Environment as a batch job. Customize the JCL and then submit it. The Execution Environment initializes and waits for VTAM login requests from terminal users.

Starting the Native Execution Environment as a Started Task

Member NATIVEEE in the JCL data set can be used to create a Native Execution Environment started task. Complete the following steps:

1. Customize the JCL procedure.
2. Modify the member as appropriate to create a started task and copy it to a system JCL procedure library.
3. Issue the z/OS **START** command from a system console.

   An example of the command required is given below. It assumes that the name of the started task is NATIVEEE:

   ```
   S NATIVEEE<Enter>
   ```

   The procedure NATIVEEE starts, and the Native Execution Environment initializes and waits for VTAM login requests from terminal users.

Log In to a Native Execution Environment

Use the EENAME= initialization parameter as the name of the Native Execution Environment you want to log in to. Two examples of the VTAM **LOGON** command are shown below. The one you use depends on how the VTAM USSTAB entries are coded:

- `LOGON APPLID(vtamapplid) DATA('U=userid,P=password')`
- `LOGON APPLID=vtamapplid DATA='U=userid,P=password'`

In these examples, the following parameters are specified:

vtamapplid	Defines the MDL parameter value specified when the Native Execution Environment was initialized.
userid	Specifies the user ID that the Native Execution Environment uses to create an Execution Environment session. There is no default. A valid TIBCO Object Service Broker user ID must be specified.
password	*Optional*: Specifies the password of the user ID that is specified by U= parameter. If P=*password* is not specified, the user is prompted for a password.

Specifying Other Session Parameters

Other session parameters can be specified using the DATA parameter of this command. VTAM restricts the length of the DATA parameter string to 64 characters.

Stop the Native Execution Environment

To stop a Native Execution Environment, use the z/OS **STOP** or **MODIFY** commands. The following shows an example of each:

```
P NATIVEEE
```

```
F NATIVEEE,SHUTDOWN
```

In these examples, *NATIVEEE* is the name of the batch job or started task used to start the Native Execution Environment.

Modify @SCHEDULEMODEL to Enable Printing

To print output generated from a Native Execution Environment, the 'MVS,SPOOLSTRIP' @SCHEDULEMODEL table instance must be modified as explained in Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table on page 70.

Chapter 5 Installing the Service Gateway for CICS

This chapter describes how to install and configure the Service Gateway for CICS.

Topics

Introduction

The Service Gateway for CICS provides an Execution Environment in a CICS environment. It makes the necessary functions available to accommodate multiple TIBCO Object Service Broker sessions in the CICS environment. By accommodating a large number of user sessions in a single address space, overhead is reduced. The Service Gateway for CICS also provides an alternative to using the Native Execution Environment (NEE) as the target for TIBCO Object Service Broker Eclipse UI sessions.

Service Gateway for CICS ensures that data is presented to TIBCO Object Service Broker rules in a manner consistent with TIBCO Object Service Broker behavior.

For details on using Service Gateway for CICS, see *TIBCO Object Service Broker for z/OS External Environments*.

Preparing for Installation

Before installing Service Gateway for CICS, review the following:

- TIBCO Object Service Broker **Base Component** – You must install and ACCEPT (using SMP/E) the TIBCO Object Service Broker base component before installing Service Gateway for CICS. You must also have the `<HLQ>.INSTALL` data set that was created during that installation.
- **Language Environment** – Within a CICS environment, you must install the CICS RDO definitions in CEE.SCEESAMP(CEECCSD) or have the CICS autoinstall be operational for programs. For additional information see Language Environment (LE) on page 4.
- **Supported Versions of CICS** – Refer to the Late Breaking News on our http://support.tibco.com/ web site for the most current information about the levels, versions, and releases of CICS that Service Gateway for CICS supports.
- **OSEMOD ISPF Edit Macro for CICS Variables** – Refer to Appendix A, CICS Variables on page 404 for more information on customizing CICS variables.

Distribution Media and Contents

This section describes how to obtain the software, and the installation file that comprises the distribution media. Similar to the TIBCO Object Service Broker base component, the Service Gateway for CICS software is distributed in `.xml` format within a ZIP file.

Distribution File Format

The file is in a format compatible with IBM System Modification Program/Extended (SMP/E) naming conventions. The product is packaged in SMP/E txlib format.

Obtaining the Installation Media

As with the TIBCO Object Service Broker base component, you can download the software from the TIBCO Software web site by following these steps:

1. Contact TIBCO Software Inc. for a password, directory information, etc.
2. Connect to the TIBCO web site with the required information.
3. Download the appropriate ZIP file.

Installation Files

The following ZIP file comprises the distribution media:

```
TIB_srvcgw-cics_6.0.0_zos.zip
```

Uploading the Software

If you have acquired the Service Gateway for CICS by downloading it from the TIBCO Software web site, you must upload the software to the z/OS host system.

Preparing the Product File for Uploading

1. Download or copy the `TIB_srvcgw-cics_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.
2. Unzip the file to a temporary location on the PC. The file contains the following:

   `cics.xm1` – compressed file containing Service Gateway for CICS.
3. Pre-allocate the following sequential data set on the z/OS host system:

   `<HLQ>.CICS.XM1 (size 22 KB)`

   Use the same <HLQ> that you specified when you uploaded the base component. Below is sample JCL to allocate this data set. Provide a JOB card and submit the JCL.

```
//ALLOC EXEC PGM=IEFBR14
//DD1   DD DSN=<HLQ>.CICS.XM1,
//      DISP=(,CATLG,DELETE),UNIT=SYSDA,
//      DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
//      SPACE=(TRK,(2,1))
```

Creating the Product Installation Data Set

To create the product installation data set, FTP the `cics.xm1` file in BIN mode to the `<HLQ>.CICS.XM1` data set.

Installing the Software

You must perform the installation under an ISPF environment.

This section describes the procedure for installing Service Gateway for CICS. These instructions are for a new installation and for migrating from a previous version.

You can start the installation if you have the following data sets ready:

- <HLQ>.INSTALL
- <HLQ>.CICS.XM1

You must use the <HLQ>.INSTALL data set that was created during the installation of the TIBCO Object Service Broker base component.

Checklist of system environment information you have to provide for this install:

- Data set name of your CICS system load library

Edit the Properties File

Edit the PROPERTY member in <HLQ>.INSTALL. Table 5 describes the keywords in the properties file for installing this component.

Table 5 Properties File Keywords

Keyword	Description
INSTALL=	To install a local Service Gateway for CICS, specify CICS: INSTALL=CICS
CICSDSN=	The fully qualified unquoted DSName of the CICS system load library; for example, CICSTS31.CICS.SDFHLOAD.

If member PROPERTY was used for another install, save it by renaming the member, and then create a new PROPERTY member with the keywords above.

Installation

STEP 1: **Execute File Tailoring EXEC to start installation.**

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

The installer verifies the value of the `CICSDSN keyword`. If valid, the installer creates the JCL to complete the pre-installation phase.

STEP 2: **Run Job CICS.JCL.**

This batch job will uncompress the `CICS.XM1` file to produce the distribution library.

JCL in: `<HLQ>.CICS.JCL (Edit the JOB card to your site's standards)`

Data Set: `<HLQ>.CICS.JCL (SUB data set)`

Uncompressing `<HLQ>.CICS.XM1` produces the distribution library `<HLQ>.CICS.FILEI.`

STEP 3: **Create and customize work copies of data sets.**

Member in: `<HLQ>.CICS.FILEI`

Member: `S6B1CUST (EX member)`

The following work copies are created and customized with values specified by OSEMOD variables:

- `<HLQNONV>.<INSTVER>.JCL` – sample JCL
- `<HLQNONV>.<INSTVER>.CICS.JOBS` – install jobs for CICS

STEP 4: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.CICS.JOBS`

Member: `S6B2RUNJ (EX member)`

SEND messages are directed to the userid specified in the NOTIFY parameter of each job submitted, informing the user of submission, and normal completion or abnormal termination. On successful completion of the final job in JOBSB list, the message ALL MEMBERS PROCESSED displays. This completes the auto-installation process for CICS.

You can modify the STATUS of any job as per your requirement. For example, if your shop normally ACCEPTs the product FMID at some future time, then change the status of S6B4ACPT from INSTALL to FUTURE. Note that you must ACCEPT the CICS component before applying any hotfix maintenance using SMP/E.

After installing Service Gateway for CICS, perform the following tasks:

- Installing and Customizing the CICS Node Error Program
- Defining Resources to CICS
- Creating Control Table Definitions
- Creating and Modifying Other CICS Definitions and Parameters
- Testing the Software

Installing and Customizing the CICS Node Error Program

Installing and customizing the DFHZNEP CICS node error program enables TIBCO Object Service Broker to automatically terminate a user session when the terminal is disconnected and to free the resources allocated to the session.

Sample Routines

The S6BZNEPC sample is provided in the ASM data set to use as a guide when customizing the CICS supplied default node error program.

If your site already has a customized DFHZNEP CICS exit installed, review the sample and implement the logic necessary to call the TIBCO Object Service Broker lost-terminal processing routine.

Error Conditions That Call the Lost Terminal Processing Routine

In the sample provided, the TIBCO Object Service Broker routine is called under one of the following conditions:

- When the “default action flags” field set by DFHZNAC (the CICS node abnormal condition program) is set to abnormally terminate or to cancel the task that is attached to this terminal
- Error code X'49' — Terminal placed out-of-service with CEMT
- Error code X'57' — Terminal released with CEMT

Consider adding other codes appropriate to your processing needs.

Link to the S6BCSNEM Module

In the appropriate node error program handlers, invoke the TIBCO Object Service Broker routine by linking to the module S6BCSNEM (refer to the sample programs for coding details). When S6BCSNEM is invoked, the following occurs:

- S6BCSNEM determines if the terminal being disconnected is running a TIBCO Object Service Broker session. This is done by retrieving a CICS temporary storage queue that is built when a TIBCO Object Service Broker session starts.
- If no record is found, TIBCO Object Service Broker returns control to the calling DFHZNEP module since no session is active.
- If a record is found, TIBCO Object Service Broker schedules a CANCELUSER operation before returning control to DFHZNEP.

CANCELUSER Operations

The CANCELUSER operation occurs in two ways, depending on the location of the canceled session:

- If the canceled session is running in the same CICS system, S6BCSNEM schedules an HCUS CANCELUSER for the associated session.
- In a Terminal Owning Region (TOR), where the TIBCO Object Service Broker session is running in another CICS system, an internal HOPT transaction starts locally. This transaction then starts an HCUS CANCELUSER in the remote CICS system running TIBCO Object Service Broker. The extra processing is due to CICS function shipping restrictions within the DFHZNEP environment.

Install the Supplied DFHZNEP Exit

To install the supplied DFHZNEP exit with SMP/E, complete the following steps:

1. Select the USERMOD4 member in the JCL data set.

   These are sample SMP/E usermods to apply a DFHZNEP module to implement TIBCO Object Service Broker CICS lost-terminal handling logic.

2. Customize USERMOD4.
3. Submit the usermod. It should end with a RC=0.

Proceed to Defining Resources to CICS on page 97.

Defining Resources to CICS

The following section specifies the CICS definitions that you must make to use TIBCO Object Service Broker. The member CICSDEFS in the JCL data set contains a copy of the definitions given below.

Customize the CICS resource definitions before loading them into your CICS system definitions file DFHCSD.

The following CICS LISTs are created by the member CICSDEFS:

$OSBNMR$	TIBCO Object Service Broker CICS single region definitions.
$OSBAOR$	TIBCO Object Service Broker Application-Owning Region (AOR).
$OSBTOR$	TIBCO Object Service Broker Terminal-Owning Region (TOR).

By removing commented statements in CICSDEFS, you can optionally install TOR or AOR connections to test multiregion CICS operations. More information is available in CICSDEFS. To implement one or more of these lists, add them to the GRPLIST CICS startup parameter.

To use the SDK (C/C++) server, you must run definitions for that component. For these additional requirements for a CICS environment, refer to *TIBCO Object Service Broker for z/OS External Environments*.

CICS transactions and default transaction names are provided with TIBCO Object Service Broker. Your site can change the TIBCO Object Service Broker CICS transaction names. The documentation below uses the default names provided with TIBCO Object Service Broker.

Do not use the CICS CEMT SET PRO(S6BCS*xxx*) NEWCOPY command when the TIBCO Object Service Broker CICS interface is active. Failure to observe this warning causes unpredictable results, including Execution Environment failure.

Initializing the Gateway

Authorizing the Service Gateway for CICS

The Service Gateway for CICS communicates with the Data Object Broker via a subtask attached during initialization. You must authorize TIBCO Object Service Broker in a CICS address space to allow the use of Cross Memory Services by installing the TIBCO Object Service Broker SVC.

TIBCO Object Service Broker must run authorized under any of the following conditions:

- Password encryption is used.
- The FORCE CANCELUSER facility (HFUS transaction) is to be used.
- SMF recording for TIBCO Object Service Broker is required.
- VTAM SRB processing is desired.

Refer to Installing the TIBCO Object Service Broker SVC on page 52 for the procedure to install the TIBCO Object Service Broker SVC. TIBCO Object Service Broker runs unauthorized under CICS if these steps are not performed.

Automating Initialization

To automate Service Gateway for CICS initialization, a program name is defined in the CICS PLTPI. This enables Service Gateway for CICS to accept TIBCO Object Service Broker session connection requests when CICS initialization is complete.

Service Gateway for CICS initialization does the following:

- Creates an Execution Environment subtask
- Establishes an Execution Environment connection to the Data Object Broker using Cross Memory Services, VTAM, or TCP/IP
- Creates control blocks and work areas

Logging In to Data Object Broker

After initialization is complete, users enter the **HURN** transaction to log in to the Data Object Broker.

HURN is the default transaction name provided with TIBCO Object Service Broker. Any valid CICS transaction name can be defined by the site in place of **HURN** as long as it does not conflict with the names selected for the other TIBCO Object Service Broker transactions. TIBCO Object Service Broker is sensitive only to the transaction names **HCUS**, **HCSS**, and **HOPT**. **HCUS** must be defined to the CICS systems running the Execution Environment. The transaction can be renamed to differentiate multiple Execution Environment regions connecting to the same Terminal Owning Region (for example, **HCUS** to **XXUS**).

Automating Shutdown

When CICS is shut down, Service Gateway for CICS can be shut down automatically by defining the TIBCO Object Service Broker CICS termination program name in the CICS PLTSD. Service Gateway for CICS shutdown does the following:

- Disconnects the Execution Environment from the Data Object Broker
- Frees control blocks and work areas
- Detaches the Execution Environment subtask

Alternatively, the Execution Environment can be shut down using **HTRM SHUT** or **HTRM SHUTI**. These are transactions supplied with TIBCO Object Service Broker.

Creating Control Table Definitions

Single CICS Region

If you are running a single CICS region, define the following programs in Resource Definition Online (RDO). You can also use the CICSDEFS member in the JCL data set to define these programs in LIST $OSTARNMR$.

Transaction ID	Transaction	Pointing to Program...
HAPI	Call Level API Test Program.	HRNCALIN
HCLI	Application Processing: Standby Session Server.	S6BCSCLI
HCSS	Internal Timeout.	S6BCSCSS
HCUS	Canceluser.	S6BCSCUS
HFUS	FORCE Canceluser.	S6BCSFUS
HINQ	Operator Inquiry.	S6BCSINQ
HINT	Operator Initialization.	S6BCSINT
HREL	HCS RELAY Operator Command Processor.	S6BCSREL
HSC1	Application Processing: Display / Seamless.	S6BCSSC1
HSN1	Application Processing: non-Display / Seamless.	S6BCSSN1
HSN2	Application Processing: non-Display / non-Seamless.	S6BCSSN2
HSRV	External Database Server Operator Control.	S6BCSSRV
HTC1	Application Processing: Display / Seamless.	S6BCSTC1
HTC2	Application Processing: Display / non-Seamless.	S6BCSTC2
HTN1	Application Processing: non-Display / Seamless.	S6BCSTN1

Transaction ID	Transaction	Pointing to Program...
HTN2	Application Processing: non-Display / non-Seamless.	S6BCSTN2
HTRM	Operator System Termination.	S6BCSHUT
HURN	Application Processing: Display / non-Seamless.	S6BCSSC2

Usage Notes

Note the following when defining the transactions:

- You must specify TASKDATAKEY(USER) and TASKDATALOC(ANY).

  Note: You can specify TASKDATAKEY(CICS) but doing so is overrestrictive, preventing users from calling CICS external routines with EXECKEY(USER). Instead, users must define their program as EXECKEY(CICS), which allows the external CICS routine, if it contains an addressing error, to corrupt CICS storage.

- TIBCO Object Service Broker is sensitive only to the specific transaction names **`HCUS`**, **`HOPT`**, and **`HCSS`**. You can assign your own names, one to four characters in length, or PF keys for the other transactions.

- If you do use your own transaction name instead of **`HURN`**, you must specify the Execution Environment parameter CICSHURONTRAN=*yourname*. This is required for proper initialization of TIBCO Object Service Broker peer servers.

- If you require a lowercase transaction name, use the ALIAS parameter.

- If you are running in a Multi-Region Option (MRO) environment or using Inter System Communications (ISC), you must include the following information so the user ID is passed to the Application Owning Region (AOR):

  ```
  ATTACHSEC(IDENTIFY) (Connection definitions)
  ```

Multi-Region Option (MRO) Environment

If you are running in an MRO environment, define these transactions to both the Terminal Owning Region (TOR) and the AOR. The Execution Environment executes within the AOR.

Setting Up an MRO Environment

To set up an MRO environment, define the TIBCO Object Service Broker transactions as being remote to the TOR and local to the AOR. For pseudo-conversational CICS environments, the transaction invoking S6BCSSC2 must use the same transaction identification for both the TOR remote definition and the AOR local definition. It is recommended that the same transaction identification be used in both the TOR and AOR, whether or not you are using pseudo-conversational processing. Refer to *TIBCO Object Service Broker Parameters* for a description of the CICSPSEUDOCONVERSE parameter.

Define TOR Transaction Definitions

Define the following TOR transaction definitions using RDO. You can also use the CICSDEFS member in the JCL data set to define these definitions in LIST $OSTARTOR$.

Transaction ID	Transaction
HAPI	Call Level API Test Program.
HCUS	Canceluser.
HFUS	FORCE Canceluser.
HINQ	Operator Inquiry.
HINT	Operator Initialization.
HOPT	Internal TOR Function Shipping Canceluser.
HREL	HCS RELAY Operator Command Processor
HSC1	Application Processing: Display / Seamless.
HSN1	Application Processing: non-Display / Seamless.
HSN2	Application Processing: non-Display / non-Seamless.
HSRV	External Database Server Operator Control.
HTC1	Application Processing: Display / Seamless.
HTC2	Application Processing: Display / non-Seamless.
HTN1	Application Processing: non-Display / Seamless.

Transaction ID	Transaction
HTN2	Application Processing: non-Display / non-Seamless.
HTRM	Operator System Termination.
HURN	Application Processing: Display / non-Seamless.

Refer to Notes on Usage on page 104 for additional information about defining the transactions.

All transactions are remote to the attached AOR except **HOPT**, which executes in the TOR and must be associated with the program S6BCSOPT. **HOPT** must not be defined as remote and it must run in the TOR.

Define AOR Transaction Definitions

Define the following AOR transaction definitions using RDO. You can also use the CICSDEFS member in the JCL data set to define these definitions in LIST $OSTARAOR$.

Transaction ID	Program
HAPI	S6BCALIN
HCLI	S6BCSCLI
HCSS	S6BCSCSS
HCUS	S6BCSCUS
HFUS	S6BCSFUS
HINQ	S6BCSINQ
HINT	S6BCSINT
HREL	S6BCSREL
HSC1	S6BCSSC1
HSN1	S6BCSSN1
HSN2	S6BCSSN2
HSRV	S6BCSSRV

Transaction ID	Program
HTC1	S6BCSTC1
HTC2	S6BCSTC2
HTN1	S6BCSTN1
HTN2	S6BCSTN2
HTRM	S6BCSHUT
HURN	S6BCSSC2

Notes on Usage

When defining the TOR and AOR transactions:

- You must specify TASKDATAKEY(USER) and TASKDATALOC(ANY).
- TIBCO Object Service Broker is sensitive only to the specific transaction names **HCUS**, **HCSS**, and **HOPT**. You can assign your own names, one to four characters in length, or PF keys for the other transactions.
- If you are running TIBCO Object Service Broker in multiple CICS AORs off the same TOR, the **HINT**, **HTRM**, **HURN**, **HINQ**, **HCUS**, and **HFUS** transactions must have different names for each AOR.
- If you require a lowercase transaction name, use the ALIAS parameter.
- When running MRO or using ISC, include the following information so the user ID is passed to the Application Owning Region (AOR):

  ```
  ATTACHSEC(IDENTIFY) (Connection definitions)
  ```

  Select a number of ISC sessions appropriate to your site. The greater the number of users you expect to be logged in to TIBCO Object Service Broker, the higher this number should be. The minimum required by TIBCO Object Service Broker is 2. The IBM default is 1.

Creating and Modifying Other CICS Definitions and Parameters

Using CICS Dynamic Storage Area Parameters

TIBCO Object Service Broker runs as a CICS key task. It makes no use of CICS user-key storage. The storage for the CICS address space is allocated according to the following CICS initialization parameters:

Parameter	Description
EDSALIM	Specifies the upper limit of the total amount of storage *above* the 16 MB line within which CICS can allocate individual extended dynamic storage areas (EDSAs). The value for this parameter depends on the number of TIBCO Object Service Broker sessions you anticipate needing for your installation. TIBCO Object Service Broker uses approximately 530 KB of the storage area limited by EDSALIM.
DSALIM	Specifies the upper limit of the total amount of storage *below* the 16 MB line within which CICS can allocate individual dynamic storage areas (DSAs). TIBCO Object Service Broker uses approximately 0.5 KB of the shared storage area limited by DSALIM.

TIBCO Object Service Broker also uses storage that is outside CICS control, even though it is within the CICS address space. Refer to *TIBCO Object Service Broker Parameters* for a description of the CICSREGIONSIZE parameter.

Storage Areas Limited by EDSALIM

From the storage size specified on the EDSALIM parameter, TIBCO Object Service Broker allocates the following extended dynamic storage areas:

Parameter	Description	TIBCO Object Service Broker Usage
EUDSA	Extended user DSA. The user-key storage for all user-key 31-bit task storage above the 16 MB line.	Not used by TIBCO Object Service Broker.
ERDSA	Extended read-only DSA. The key-0 storage area for all reentrant 31-bit programs and tables above the 16 MB line.	500 KB—for loading CICS-related TIBCO Object Service Broker modules that are related to CICS. These are permanently resident and persist as long as CICS is running.
ESDSA	Extended shared DSA. The user-key storage area for non-reentrant user-key RMODE(ANY) programs. Also for storage obtained by programs issuing CICS GETMAIN commands for shared storage above the 16 MB line.	30 KB of shared storage for major control blocks needed for the duration of each TIBCO Object Service Broker session under CICS storage
ECDSA	Extended CICS DSA. The CICS-key storage area for the following: All non-reentrant CICS-key RMODE(ANY) programs All CICS-key task-lifetime storage above the 16 MB line CICS control blocks that reside above the 16 MB line	3 KB—released as soon as the CICS transaction terminates

Storage Areas Limited by DSALIM

From the storage size specified on the DSALIM parameter, TIBCO Object Service Broker allocates the following dynamic storage areas:

Parameter	Description	TIBCO Object Service Broker Value
UDSA	User DSA. The user-key storage for all user-key 31-bit task storage below the 16 MB line.	Not used by TIBCO Object Service Broker.
RDSA	Read-only DSA. The key-0 storage area for all reentrant 31-bit programs and tables below the 16 MB line.	Not used by TIBCO Object Service Broker.
SDSA	Shared DSA. The user-key storage area for non-reentrant user-key RMODE(24) programs. Also for storage obtained by programs issuing EXEC CICS GETMAIN commands for shared storage below the 16 MB line.	0.5 KB of shared storage used for the duration of each TIBCO Object Service Broker session
CDSA	CICS DSA. The CICS-key storage area for the following: All non-reentrant CICS-key RMODE(24) programs All CICS-key task-lifetime storage below the 16 MB line CICS control blocks that reside below the 16 MB line	Not used by TIBCO Object Service Broker

See Also

The IBM manual *CICS Transaction Server for* z/OS *System Definition Guide* for more information about CICS storage areas and other CICS initialization parameters.

The IBM manual *CICS Transaction Server for* z/OS *Performance Guide* for information about calculating the amount of storage to specify on the EDSALIM and DSALIM parameters.

Create Processing Program Table (PPT) Definitions

Define the following TIBCO Object Service Broker programs using RDO. For either a standalone region or an Application-Owning Region (AOR), you can also use the CICSDEFS member in the JCL data set to create a group to contain these definitions. You must specify EXECKEY(CICS) and DATALOCATION(ANY).

Program	Resident?
S6BCSCLI	Y
S6BCSCSS	Y
S6BCSCUS	Y
S6BCSFUS	Y
S6BCSHUT	Y
S6BCSIDU	
S6BCSINQ	Y
S6BCSINT	Y
S6BCSKRN	Y
S6BCSNEM	
S6BCSSC1	Y
S6BCSSC2	Y
S6BCSSN1	Y
S6BCSSN2	Y
S6BCSSRV	Y
S6BCSTC1	Y
S6BCSTC2	Y
S6BCSTN1	Y
S6BCSTN2	Y

Program	Resident?
S6BCSTRM	
S6BCALIN	

Do not specify the RES=YES option for the following programs:

- S6BCSIDU
- S6BCSNEM
- S6BCSTRM
- S6BCALIN

The value ANY for TASKDATALOC and DATALOCATION is not required in all situations. ANY is recommended to take advantage of CICS Virtual Storage Constraint Relief (VSCR). In a CICS environment where TIBCO Object Service Broker invokes (or is invoked by) an AMODE(24) program, the value for these parameters *must* be BELOW.

Create Definitions for MRO Terminal Owning Regions

Define the S6BCSNEM and S6BCSOPT programs in MRO terminal-owning regions (both programs are non-resident). For a Terminal-Owning Region (TOR), you can also use the CICSDEFS member in the JCL data set to create a group to contain definitions for these programs and TIBCO Object Service Broker TOR transactions.

Create Program List Table (PLT) Startup Definitions

Use the following as a guide to define the TIBCO Object Service Broker startup programs in the CICS Program List Table (PLT), or the equivalent in RDO:

```
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=S6BCSINT
```

In this example, the **DFHDELIM** CICS program must precede the **S6BCSINT** TIBCO Object Service Broker program.

Create Program List Table Shutdown Definitions

Use the following as a guide to define the TIBCO Object Service Broker shutdown program in the CICS PLT, or the equivalent in RDO:

```
DFHPLT TYPE=ENTRY,PROGRAM=S6BCSTRM
```

```
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
```

In this example, the **S6BCSTRM** TIBCO Object Service Broker program must precede the **DFHDELIM** CICS program.

Create Transaction List Table (XLT) Definitions

Use the following as a guide to define a list of TIBCO Object Service Broker transactions that can be executed during the first quiesce state of CICS termination in the CICS XLT, or the equivalent in RDO:

```
DFHXLT TYPE=ENTRY,
TRANSID=(HCSS,HCUS,HFUS,HINQ,HTRM,HSRV,hcus,hfus,hinq,htrm,hsrv)
```

Be sure to include transactions defined in your control table definitions (see Modify Terminal Control Table (TCT) Definitions on page 110, and sections following) in the set of transaction identifications, that is, **HCSS**, **HCUS**, **HFUS**, **HINQ**, **HTRM**, and **HSRV** in this example. Transactions allowing new TIBCO Object Service Broker sessions to be initiated should not be included in this list because regular sessions are being terminated during CICS termination.

Modify Terminal Control Table (TCT) Definitions

Terminals with mixed-case characters displayed must have their Terminal Control Table (TCT) definitions modified to include the UCTRAN parameter. If not done, text data is saved and reappears in uppercase only. You must specify UCTRAN(NO) in all terminal TYPETERM definitions where mixed case characters must be input and displayed. Before making this change, consider the following:

- Some existing applications require this parameter to be YES.
- If NO is specified, the user must type the transaction ID in uppercase (for example, **HURN**) or CICS does not recognize the transaction as valid. Use the ALIAS parameter to define lowercase versions of each transaction identification.
- In other applications, users could actually save records in lowercase, causing problems with key searches (for example, part~=PART).

Define a CICS Console

When using a TIBCO Object Service Broker CICS region to start a TIBCO Object Service Broker server, the CICS console facility is used to initiate the required non-terminal background CICS tasks. These tasks provide CICS services to the server. To use the console, you must define a CICS console definition. The values for the definition depend on the version of CICS you are using. Sample console definitions appear in the S6BTERM1 group of the CICSDEFS member in the JCL data set.

Make sure you define sufficient RACF authority to your CICS region's security identifier to allow TIBCO Object Service Broker to issue modify commands internally to CICS.

The console modify facility is also used in cancelling CICS sessions from the Data Object Broker region.

See Also

The *IBM CICS Transaction Server for* z/OS *Resource Definition Guide* documentation for more information about the use of the resource definition facility.

Specify System Initialization Table (SIT) Parameters

The following parameters should be specified in the CICS System Initialization Table (SIT) or via CICS SIT override facilities:

PLTPI=*xx*	Substitute the initialization PLTPI suffix for *xx*.
PLTSD=*yy*	Substitute the shutdown PLTSD suffix for *yy*.
XLT=*zz*	Substitute the appropriate Transaction List Table for *zz*. Refer to Create Transaction List Table (XLT) Definitions on page 110 for details.

Customize Parameters

Sample initialization parameters used by TIBCO Object Service Broker in the CICS address space are provided in member PARMCICS of the CNTL data set. Step CICS in member EECONFIG in the JCL data set can be used to create a default option module for the CICS environment. Customize the initialization parameters and run the step to create the default option module.

During CICS TIBCO Object Service Broker initialization, the SVC (Supervisor Call) number is loaded from S6BDRCC0 regardless of the CONFIGURATION= specification. For a description of the sample initialization parameters included in this step, refer to *TIBCO Object Service Broker Parameters*.

Include the Authorized TIBCO Object Service Broker Data Set in Concatenations

TIBCO Object Service Broker modules are installed in an authorized data set. Include this data set in the CICS STEPLIB and DFHRPL concatenations. For more information on making the load library authorized, refer to Configuring Authorized Libraries on page 22.

Testing the Software

Starting the Service Gateway for CICS

CICS PLTPI is used to start the Service Gateway for CICS automatically at CICS startup; however, you can start Service Gateway for CICS independently of CICS startup using the **HINT** transaction. You can run it from a user terminal or system console. Valid parameters for **HINT**, which can be specified in any order, are as follows:

Parameter	Description
INSTLIB	Defines the name of the installation library. It is the TIBCO Object Service Broker library to be searched after the local library and before the system library (depending on the search path) to locate TIBCO Object Service Broker rules.
MDL	Defines the pattern to allocate an Execution Environment communications identifier.
SYSLIB	Defines the system library. It is the library to be searched after the installation library (depending on the search path) to locate rules.
TDS	Defines the communications identifier of the Data Object Broker you want to connect to.

A sample invocation of the **HINT** transaction is:

```
HINT TDS=EP01SRV,INSTLIB=BANK03
```

Shutting Down the Service Gateway for CICS

CICS PLTSD is used to shut down the Service Gateway for CICS automatically at CICS shutdown. To perform an orderly shutdown, complete the following steps:

1. Use the **HINQ** transaction to determine the number of users logged in to TIBCO Object Service Broker.
2. Wait until all the users are signed off the system and then use the regular CICS procedure to complete the system shutdown.

Alternatively, use the **HTRM SHUT** or **HTRM SHUTI** transaction to shut down Service Gateway for CICS without shutting down CICS.

Enabling Printing From CICS

To print output from CICS, modify the @SCHEDULEMODEL(MVS,SPOOLSTRIP) table instance as explained in Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table on page 70.

Do *not* use the name S6BDRPRT for a print destination in the JES complex. S6BDRPRT is the name of the offline print utility that handles the generation and printing of output from the Native Execution Environment and CICS. It runs the SPOOLSTRIP job, which does not operate correctly if the print destination is set to S6BDRPRT.

CICS Installation Verification Procedure

Overriding Default Parameters

Your initialization parameters are defined in CNTL(PARMCICS) and are instantiated by executing the JCL(EECONFIG) job, step CICS. You can override parameters at the CICS job level by including a DD statement in your CICS job step, such as the following:

```
//HRNIN DD *
* place your override options here ...
*
* define communications id for your CICS region EE
* make sure to add to CNTL(RELAYCFG) HCS RELAY configuration
EENAME=eeeeeeee,  /* should not end with a numeric */
STANDBYNUM=3,     /* standby sessions to start for Eclipse */
TDS=$TDS$         /* your DOBs communication id */
/*
```

Customizing and Executing the CICS Environment

The CICS Environment can be executed as a batch job or a started task. CICS must execute from authorized data sets. For more information on making the load library authorized, see Configuring Authorized Libraries on page 22.

You must add the TIBCO Object Service Broker load library to the CICS STEPLIB and DFHRPL DD statement concatenations. Make sure that the DFHPLT startup and shutdown tables are customized as well as all CICS resources (transactions, profiles, programs, and so on) defined to your CICS system. For more information, see Creating Control Table Definitions on page 100. DFHZNEP must be customized to include Object Service Broker lost terminal error handling. For details. see Installing and Customizing the CICS Node Error Program on page 95.

To connect the Eclipse User Interface (UI) to your CICS system, you will need to review the CICS requirements listed in Chapter 18, Introducing TIBCO Object Service Broker SDK (C/C++) Server, in *TIBCO Object Service Broker for z/OS External Environments*. Make sure that the EENAME parameter chosen for your CICS system is defined in the TCP/IP section of the HCS relay file, CNTL(RELAYCFG). For details, see The Relay File on page 424.

Running the Installation Verification Procedure

To run the IVP, perform the following:

1. Bring up your CICS system in the usual way after customizing CICS for TIBCO Object Service Broker.
2. Using a VTAM defined terminal or terminal emulator, connect to your CICS system as per your usual connection instructions. Depending on your VTAM USSTAB entries, the logon sequence would be something like the following:

   ```
   LOGON APPLID(CICSapplid), or
   LOGON APPLID=CICSapplid
   ```

   If your CICS system uses operator signon, you may have to use the CICS signon transaction CESN to identify yourself before proceeding.
3. If you have defined TIBCO Object Service Broker in your startup PLT list, the Execution Interface will have been started automatically. Issue the CICS transaction:

   ```
   HINQ
   ```

   to obtain the current status of the CICS Execution Environment interface. The result will be something like the following:

   ```
   S6BDR021I EXECUTION ENVIRONMENT IS ATTACHED - S6ELDOBA
   S6BCS022I      7 user(s).
   S6BCS052I  USER ID   TERM ID           USER ID   TERM ID
   S6BCS050I  U0043000  T0043000          $0043000  $0043000
   S6BCS050I  U0043001  T0043001          $0043001  $0043001
   S6BCS050I  U0043002  T0043002          $0043002  $0043002
   S6BCS050I  U0043003  T0043003
   ```

   In this case, the EE is running and has seven standby sessions available. These can be used to support your Eclipse User Interface sessions.

   If the CICS EE interface is not active, you will see a response to HINQ such as:

   ```
   S6BDR020I EXECUTION ENVIRONMENT IS NOT ATTACHED
   ```

   In this case you can start the interface by issuing the CICS transaction:

   ```
   HINT
   ```

   This transaction can have override parameters, such as TDS=, which identifies the Data Object Broker's communication identifier to be connected when the EE initializes. If no parameters are provided, the EECONGIF defaults will be used for the CICS region type. Response from the HINT transaction will be:

   ```
   S6BCS014I TIBCO(r) Object Service Broker/CICS interface
   initialization complete.
   ```

A message similar to the following:

```
S6BDR136I Execution Environment initialization complete,
Version: V600
```

will be displayed on the CICS JES joblog.

4. Logon via a native CICS terminal session to the TIBCO Object Service Broker. Enter your CICS defined transaction to invoke the TIBCO Object Service Broker. Typically, this will be HURN. Enter your user identification if you are not using the CESN default userid that you logged on to CICS with earlier. For example:

   ```
   HURN U=EZL30
   ```

   You will be prompted for your password if required. If you successfully logon to the EE, should be presented with the TIBCO Object Service Broker workbench. Press PF12 to terminate your EE session and return to native CICS. You should receive a message such as the following:

   ```
   S6BCS047I SESSION ENDED
   ```

   when your session terminates.

5. To check the printing capability of your CICS EE, logon to the EE using HURN U=SYSADMIN or another userid that can update the system table @SCHEDULEMODEL. Edit the table as shown below:

   ```
   ED Edit Table                ==> @SCHEDULEMODEL
   ```

   Type the table name to the right of the ED Edit Table ==> literal and press ENTER. You will be prompted for the instances of the table to edit:

   ```
   EDITING TABLE    :    @SCHEDULEMODEL

   ENTER PARM VALUE OPERATING_SYSTEM: MVS
   ENTER PARM VALUE MODELNAME       : SPOOLSTRIP
   ENTER PARM VALUE LOCATION        :
   ```

   Alter the table contents so that the JCL is acceptable to your CICS and z/OS environment. You may need to change the JCL wherever {USERID} appears, especially on the JOB card as some TIBCO Object Service Broker user identifiers may not be valid to your external security system. SYSADMIN may be one example. For information on customizing the @SCHEDULEMODEL table, see Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table on page 70.

   When you have completed the customization of the @SCHEDULEMODEL table instance MVS,SPOOLSTRIP, save the updates by pressing PF3.

   To test the changes, enter Edit of the @SCHEDULEMODEL MVS,SPOOLSTRIP instance again and check that your changes are still there. Now press PF13 to print the table instance, then press PF3 to exit Edit. Check

your JES log or your printer for the output. The JES log will contain a message such as the following:

```
$HASP100 <jobname> ON INTRDR      <jobcard identification>
FROM JOB##### <cicsjobname>
```

6. To check your ability to connect to the TIBCO Object Service Broker CICS EE, install the Eclipse User Interface (UI) on an Open Systems platform. Ensure that you have set the following EECONFIG or HRNIN override parameters for your CICS system:

   ```
   STANDBYNUM=n,
   EENAME=yourEEcommunicationsname,
   ```

   and also define `yourEEcommunicationsname` in the TCP/IP section of the HCS relay file. For details, see Customizing and Executing the CICS Environment on page 115.

   Start Eclipse and select a suitable workspace location on your drive if prompted. Then select **Window > Open Perspective > Other > OSB Perspective**. Open a new project for your CICS session by selecting **File > New >OSB Project** and specify the following:

   — Project Name – a meaningful description of the project, such as “Test CICS Eclipse UI for 6.0."

   — Host – name of system where CICS is executing, such as sandbox.your.company.com or IP address.

   — Port – the Execution Environment listening port identified in the CICS EE startup message, such as the following:

   ```
   S6BRA008I TCP/IP RELAY started; Accepting connections on
   socket port 10061
   ```

   — Userid and password to connect and start a session within the TIBCO Object Service Broker Execution Environment

   — Library – default library used by this userid/project.

   Once these have been specified, press the **Test Connection** button to see if the physical connection works. If so, you will receive the message:

   ```
   Connection test successful.
   ```

   otherwise, you might receive a message such as:

   ```
   Cannot open connection to the server.
   ```

   If you have a failure, review the settings in your profile against your z/OS and CICS specifications. Check the CICS JES and error logs for error messages. Make sure you have configured your SDK support.

Once the connection test is successful, press **Finish** to begin your Eclipse session with TIBCO Object Service Broker. To make sure that data can be retrieved, hover your cursor over Tables under OSB Projects and right mouse click, selecting **Show Tables**. This should retrieve a list of tables defined to your system. If no list is displayed, check the CICS JES joblog and error logs for failure messages if no reason is supplied by your Eclipse session.

If a list of tables is displayed. Your Eclipse OSB UI is working successfully. You may then select **Exit Eclipse** to terminate your session and Eclipse.

7. To test the lost terminal processing under DFHZNEP, you will need to customize and stage the CICS DFHZNEP module according to the instructions mentioned previously. Start a native VTAM session with your CICS region, then start a TIBCO Object Service Broker session by entering the following:

   ```
   HURN U=your_Object_Service_user_id
   ```

   and respond with your password if prompted. When the workbench is displayed, terminate your VTAM session by disconnecting the session to drive the lost terminal processing in CICS. If DFHZNEP has been correctly customized, you should see a message on the CICS JES joblog such as the following:

   ```
   S6BCS076I <tttt>/<userid> Session cancelled by operator
   command
   ```

   On the CICS Transient Data error log, you should also see a message of the form:

   ```
   S6BCS100I S6BCSNEM started HCUS <tttt><netname> routed to
   <CICS_applid>
   ```

   The Data Object Broker JES joblog should also show a message similar to the following:

   ```
   S6BKC017L <userid> LOGGED OFF #=nnnnnn TYPE=CICSUSER
   COMM=TCP/IP SOURCE=<tttt>
   ```

8. To test out the automated PLT shutdown of the Execution Environment, logon to CICS using a 3270 session and issue the following command:

   ```
   CEMT P SHUT
   ```

   You should see on the CICS JES joblog the following messages if the shutdown is successfully executed for the TIBCO Object Service Broker EE:

```
DFHTM1715 <CICS-jobname> CICS is being quiesced by userid
<userid> in transaction CEMT at netname <nnnnnnnn>.
DFHTM1709I <CICS-jobname> About to link to PLT programs.
. . .
S6BDR078I COMMAND 'SHUTI ' ISSUED
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPPOSIX
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPINTRP
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPSIN
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPOPER
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPSORT
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPMISC
S6BDR065I TERMINATION COMPLETE FOR SERVER CLASS DSPFILIO
S6BRA009I TCP/IP RELAY stopped
S6BTK002I TRACKER Task termination complete
DFHTM1710I <CICS-jobname> Control returned from PLT programs.
```

This concludes the basic installation verification procedure for CICS Execution Environments.

Chapter 6 Installing the Service Gateway for IMS TM

This chapter describes how to install the Service Gateway for IMS TM.

Topics

Introduction

The Service Gateway for IMS TM enables IMS online users to access data from a Data Object Broker and its attached servers, and to access VSAM and sequential files from a Native Execution Environment.

Service Gateway for IMS TM provides full TIBCO Object Service Broker functionality in an IMS TM environment and is started in an IMS Message Processing Region (MPR).

Relation to Native Execution Environment

The Native Execution Environment is a component of TIBCO Object Service Broker and must be installed before or concurrently with Service Gateway for IMS TM. The Native Execution Environment is a multiple-user address space that provides all Execution Environment services without the requirement for an external teleprocessing monitor to handle communications between the application and the user.

Important Notes for Release 6.0:

- Switched from Physical Terminal (Input) Edit Routine (`DFSPIXT0`) to Global Physical Terminal (Input) Edit routine (`DFSGPIX0`)

  `<HLQNONV>.<INSTVER>.JCL(USERMOD5)`

  This provides the necessary control statements to implement the routine and the routine no longer needs to be linked in with the IMS TM virtual nucleus.

See Also *TIBCO Object Service Broker for z/OS External Environments* for more information about IMS TM clients operating in a Native Execution Environment.

TIBCO Object Service Broker Messages With Identifiers for messages produced by the IMS TM Execution Environment.

Preparing for Installation

Before installing Service Gateway for IMS TM, review the following:

- TIBCO Object Service Broker **Base Component** – You must install the TIBCO Object Service Broker base component before installing Service Gateway for IMS TM. You must also have the <HLQ>.INSTALL data set that was created during that installation.
- **Supported Versions of IMS** – Refer to the Late Breaking News on our http://support.tibco.com/ web site for the most current information about the levels, versions, and releases of IMS that Service Gateway for IMS TM supports.

Distribution Media and Contents

This section describes how to obtain the software, and the installation file that comprises the distribution media. Similar to the TIBCO Object Service Broker base component, the Service Gateway for IMS TM software is distributed in `.xml` format within a ZIP file.

Distribution File Format

The file is in a format compatible with IBM System Modification Program/Extended (SMP/E) naming conventions. The product is packaged in SMP/E txlib format.

Obtaining the Installation Media

As with the TIBCO Object Service Broker base component, you can download the software from the TIBCO Software web site by following these steps:

1. Contact TIBCO Software Inc. for a password, directory information, etc.
2. Connect to the TIBCO web site with the required information.
3. Download the appropriate ZIP file.

Installation Files

The following ZIP file comprises the distribution media:

```
TIB_srvcgw-imstm_6.0.0_zos.zip
```

Uploading the Software

If you have acquired the Service Gateway for IMS TM by downloading it from the TIBCO Software web site, you must upload the software to the z/OS host system.

Preparing the Product File for Uploading

1. Download or copy the `TIB_srvcgw-imstm_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.
2. Unzip the file to a temporary location on the PC. The file contains the following:

   `imstm.xm1` – compressed file containing Service Gateway for IMS TM.
3. Pre-allocate the following sequential data set on the z/OS host system:

   `<HLQ>.IMSTM.XM1` `(size 22 KB)`

   Use the same <HLQ> that you specified when you uploaded the base component. Below is sample JCL to allocate this data set. Provide a JOB card and submit the JCL.

```
//ALLOC EXEC PGM=IEFBR14
//DD1   DD DSN=<HLQ>.IMSTM.XM1,
//      DISP=(,CATLG,DELETE),UNIT=SYSDA,
//      DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
//      SPACE=(TRK,(2,1))
```

Creating the Product Installation Data Set

To create the product installation data set, FTP the `imstm.xm1` file in BIN mode to the `<HLQ>.IMSTM.XM1` data set.

Installing the Software

You must perform the installation under an ISPF environment.

This section describes the procedure for installing Service Gateway for IMS TM. These instructions are for a new installation and for migrating from a previous version.

You can start the installation if you have the following data sets ready:

- <HLQ>.INSTALL
- <HLQ>.IMSTM.XM1

You must use the <HLQ>.INSTALL data set that was created during the installation of the TIBCO Object Service Broker base component.

Checklist of system environment information you have to provide for this install:

- Data set name of your IMS RESLIB library

Edit the Properties File

Edit the PROPERTY member in <HLQ>.INSTALL. Table 6 describes the keywords in the properties file for installing this component.

Table 6 Properties File Keywords

Keyword	Description
INSTALL=	To install a local Service Gateway for IMS TM, specify IMSTM: INSTALL=IMSTM
RESLIB=	The fully qualified unquoted DSName of IMS RESLIB; for example, IMS1010.IMS1.SDFSRESL.

If member PROPERTY was used for another install, save it by renaming the member, and then create a new PROPERTY member with the keywords above.

Initial Installation

STEP 1: **Execute File Tailoring EXEC to start installation.**

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

The installer verifies value of the `RESLIB keyword`. If valid, the installer creates the JCL to complete the pre-installation phase.

STEP 2: **Run Job IMSTM.JCL.**

This batch job will uncompress the `IMSTM.XM1` file to produce the distribution library.

JCL in: `<HLQ>.IMSTM.JCL (Edit the JOB card to your site's standards)`

Data Set: `<HLQ>.IMSTM.JCL (SUB data set)`

Uncompressing `<HLQ>.IMSTM.XM1` produces the distribution library `<HLQ>.IMSTM.FILEI.`

STEP 3: **Create and customize work copies of data sets.**

Member in: `<HLQ>.IMSTM.FILEI`

Member: `S6G1CUST (EX member)`

The following work copies are created and customized with values specified by OSEMOD variables:

- `<HLQNONV>.<INSTVER>.JCL` – sample JCL
- `<HLQNONV>.<INSTVER>.IMSTM.JOBS` – install jobs for IMSTM

STEP 4: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.IMSTM.JOBS`

Member: `S6G2RUNJ (EX member)`

SEND messages are directed to the userid specified in the NOTIFY parameter of each job submitted, informing the user of submission, and normal completion or abnormal termination. On successful completion of the final job in JOBSG list, the message ALL MEMBERS PROCESSED displays. This completes the auto-installation process for IMSTM.

You can modify the STATUS of any job as per your requirement. For example, if your shop normally ACCEPTs the product FMID at some future time, then change the status of S6C4ACPT from INSTALL to FUTURE. Note that you must ACCEPT the IMS/TM component before applying any hotfix maintenance using SMP/E.

After installing Service Gateway for IMS TM, perform the following:

- Install IMS TM NCT Support for Global Physical Terminal (Input) Edit Routine
- Install IMS TM NCT Terminal Support for Sign-on Exit
- Install the IMS TM User-Replaceable Session Exit Routine
- Required System Changes
- Specifying Initialization Parameters
- Testing the Software.

Install IMS TM NCT Support for Global Physical Terminal (Input) Edit Routine

This step must be completed only if user sessions are to be non-conversational in MFS-bypass mode. Note the following:

- Sample member S6BDCPE1 in the ASM data set specifies the resume-transaction name when a user session is non-conversational in MFS-bypass mode. This is the only way to specify the resume-transaction name.
- If your site already has a customized DFSGPIX0 exit installed or an equivalent exit, review the sample code and implement the logic necessary to call TIBCO Object Service Broker IMS TM NCT support for global physical terminal input edit handling.
- The link-edited load module is called DFSGPIX0 as installed.

Installation

Complete the following steps:

1. Select the USERMOD5 member in data set JCL as a sample SMP/E usermod to apply a DFSGPIX0 module that implements TIBCO Object Service Broker IMS TM NCT support for the global physical terminal input edit routine.
2. Customize USERMOD5.
3. Submit the usermod. It should end with RC=0.
4. Using IEBCOPY, copy the global physical terminal (input) edit routine DFSGPIX0 into either the IMS.SDFSRESL library or another PDS data set concatenated to STEPLIB. This library must be a PDS data set. If the library is a PDSE or the module is not present in STEPLIB, the IMS Transaction Manager control region will not load the routine nor issue any error message.

5. Modify the IMS Transaction Manager control region started task JCL to include the TIBCO Object Service Broker supplied $HLQNONV$.$INSTVER$.AUTH data set in the STEPLIB concatenation in the IMS control region.

```
//STEPLIB DD DISP=SHR,DSN=IMS.SDFSRESL
//        DD DISP=SHR,DSN=customer.pds.containing.DFSFLGX0
...
//        DD DISP=SHR,DSN=$HLQNONV$.$INSTVER$.AUTH
...
```

Install IMS TM NCT Terminal Support for Sign-on Exit

Sample member S6BDCSGN in the ASM data set supports IMS TM NCT sign-on and sign-off processing. This exit *must* be installed if you want to take advantage of the following:

- Extended data stream attributes (for example, color, beeping)
- Terminal screen sizes other than model 2 (that is, models 3, 4, and 5)
- Support for CANCELUSER

Sign-on Exit Requirements

- The sign-on exit records active users in the IMS TM environment and gathers terminal screen size information. A disconnected terminal invokes the sign-off exit, causing a CANCELUSER request to terminate the disconnected user.
- Enforced sign-on must be enabled for the sign-on exit to be invoked.
- Make sure to define sufficient RACF authority to your IMS region's environmental security identifier to allow TIBCO Object Service Broker to issue modify commands internally to z/OS to cancel IMS sessions.
- If your site already has a customized DFSCSGN0 exit installed or an equivalent exit, review the sample code and implement the logic necessary to call TIBCO Object Service Broker IMS TM NCT support for sign-on and sign-off exit processing.
- The TIBCO Object Service Broker load module must be link edited into the IMS nucleus during IMS SYSGEN.

Installation

Complete the following steps:

1. Select member USERMOD6 in JCL as a sample SMP/E usermod to apply a DFSCSGN0 module that implements TIBCO Object Service Broker IMS TM NCT for sign-on and sign-off exit processing.
2. Customize USERMOD6.
3. Submit the usermod. It should end with RC=0.

Install the IMS TM User-Replaceable Session Exit Routine

You can use the session exit routine (refer to sample member S6BDCUSX in the ASM data set) to bypass some processing in the TIBCO Object Service Broker IMS TM client program or conversational continuation program and to provide substitute values for some of these indicators:

- The exit phase
- The interface style (seamless, conversational, user ID derivation)
- MODname and trancode
- PCB parameter list
- Address of input/output message segments

Refer to *TIBCO Object Service Broker for z/OS External Environments* for information on using this exit.

Installation

Complete the following steps:

1. Select member USERMOD9 in JCL as a sample SMP/E usermod to apply this module.
2. Customize USERMOD9.
3. Submit the usermod. It should end with RC=0.

Required System Changes

The following system changes are required to allow TIBCO Object Service Broker to run.

- Allocate the TIBCO Object Service Broker Load Library
- Define TIBCO Object Service Broker IMS Transactions
- Sample Transactions Defined to IMS
- Create PSBs
- Build ACBs
- Authorize Library
- Modify and Compile the Sample IMS Logon Format Panel
- Enable Printing from IMS TM

Allocate the TIBCO Object Service Broker Load Library

The TIBCO Object Service Broker load library must be allocated in the startup JCL for the IMS TM Control Region. Use one of the following methods:

- Include it as the STEPLIB data set.
- Concatenate it to the existing STEPLIB statement and, if the STEPLIB already points to an APF non-authorized library, include a HRNLIB DD statement that also points to the TIBCO Object Service Broker load library.

The TIBCO Object Service Broker load library must be APF authorized.

See Also *TIBCO Object Service Broker for z/OS External Environments* for a definition of the HRNLIB DDname.

Configuring Authorized Libraries on page 22 for information on making the load library authorized.

Define TIBCO Object Service Broker IMS Transactions

A number of TIBCO Object Service Broker transactions must be defined to IMS. Sample definitions are shown below. Member IMSDCTRN in the ASM data set provides a copy of these definitions.

SPA Size and Transaction Class

Some of the default definitions contain a *scratch pad area* (SPA) size of 4096. This value must be a minimum of 14 bytes, and the maximum depends on the SPA size of the transactions with which TIBCO Object Service Broker interacts. TIBCO Object Service Broker itself does not update any space within the SPA. Specify a value appropriate for your site.

A transaction class of 2 is specified by default. Specify a transaction class appropriate for your site.

Mixed Case Data Input

Customers who require data input at IMS TM terminals to be saved in uppercase and lowercase must specify EDIT=ULC on the TRANSACT macro to avoid possible data translation problems.

Sample Transactions Defined to IMS

The definitions in sample member IMSDCTRN fall into three basic categories:

- The first section defines the conversational and non-conversational transactions that resume TIBCO Object Service Broker in an MPR.
- The second section defines PSB names that start with S6BIMS. These definitions use the IMS TM environment ID as the TIBCO Object Service Broker login ID.
- The last section defines PSB names that start with S6BIMT. These definitions use the IMS TM environment transaction ID as the TIBCO Object Service Broker login ID.

Create PSBs

Create PSBs to define each TIBCO Object Service Broker program to run in the IMS MPR. Refer to member IMSDCPSB in the ASM data set for a source copy of these definitions.

Build ACBs

Run the IMS utility to build the ACBs, using the PSBs just created.

Authorize Library

The following are required to use Cross Memory Services:

- The IMS TM Execution Environment modules must reside in an authorized library.
- The Supervisor Call (SVC) parameter must be specified in the PARMDC IMS TM Execution Environment initialization parameter member in the CNTL data set.
- The PARMDC member must be assembled and link edited into an option module by the EECONFIG job in JCL data set.
- The TIBCO Object Service Broker SVC must be installed and the MPR's STEPLIB DD should be authorized.

  If you want to run with Cross Memory Services and the MPR's STEPLIB DD concatenation is not authorized, the TIBCO Object Service Broker load library must be allocated in the MPR's JCL with a HRNLIB DD statement.

Modify and Compile the Sample IMS Logon Format Panel

The XIDCMFS0 member in the CNTL data set provides a sample TIBCO Object Service Broker IMS logon panel defined using IMS Message Format Services. Modify the sample as required and compile it with the Message Format Service utilities provided with your IMS system. Store it in your IMS FORMAT library. To use the sample logon panel, type:

```
/FOR HURON:
```

After entering this command, the screen named TIBCO Object Service Broker appears. The user is then able to type in their TIBCO Object Service Broker user ID, password, and session parameters.

Enable Printing from IMS TM

To print output generated from an IMS TM Execution Environment, modify the @SCHEDULEMODEL(MVS,SPOOLSTRIP) table instance as explained in Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table on page 70.

Specifying Initialization Parameters

Service Gateway for IMS TM initialization parameters can be specified in two ways:

- By specifying them in a default option module
- By coding parameters in a data set allocated to the MPR with the HRNIN DD statement

Default Environment Option Module

Member PARMDC in the CNTL data set provides sample Service Gateway for IMS TM initialization parameters. Step IMSDC in EECONFIG must be uncommented and run to create the IMS TM default environment option module.

For a description of the sample initialization parameters included in this step, refer to *TIBCO Object Service Broker Parameters*. These parameters are used during initialization to establish Execution Environment defaults.

HRNIN DD Statement

The second method of specifying parameters is to include an HRNIN DD statement in the IMS TM MPR JCL. The data set allocated by the DD contains initialization parameters in source format. Code one parameter per line. The data set must be LRECL=80 and RECFM=FB. Parameters specified by the HRNIN DD override the default values established during Execution Environment initialization.

Initialization Parameters

The following parameters are unique to Service Gateway for IMS TM. See *TIBCO Object Service Broker Parameters* for a description of these parameters.

EENAME=	Specifies which Native Execution Environment the MPR is to communicate with. The use of the EENAME parameter is different from the other TIBCO Object Service Broker servers, since the Native Execution Environment is the object of session initiation requests (from other Execution Environments) and an initiator of session requests (to the Data Object Broker).

IDPREFIX=	Specifies the IMSID of the IMS TM Data Object Broker to which the Execution Environment is connected. The value of the IMSID is a 1 to 4 character name that uniquely identifies the IMS TM Data Object Brokers. Within TIBCO Object Service Broker applications the value can be retrieved by the $GETOPT tool.
IMSSCREENATTRIBU= YES/NO	Specifies whether terminals with sessions established under a given MPR/Native Execution Environment have extended data stream support. If this parameter is not specified in the MPR, the setting in the Native Execution Environment is used. If conflicting values are specified, the MPR overrides the Native Execution Environment. There is no default. If a value is not specified either in the MPR or the Native Execution Environment then extended data stream is not supported.
IMSSCREENTRAN= S6BDCKRN	Specifies the name of the TIBCO Object Service Broker IMS conversational transaction to invoke when resuming a TIBCO Object Service Broker IMS TM transaction within an MPR. To run with a different version of S6BDCKRN (for testing purposes), specify a transaction code name on the IMSSCREENTRAN parameter that is different from the default S6BDCKRN. To avoid an 806 abend when loading the module with the new name, copy the test version of S6BDCKRN with a new name (same as the Transaction Code) into the load library. The PSB name must match the transaction code to avoid a 0C4 abend.
IMSSCREENTRANNC= S6BNCKRN	Specifies the name of the TIBCO Object Service Broker IMS non-conversational transaction to invoke when resuming a TIBCO Object Service Broker IMS TM transaction within an MPR.
REGIONTYPE= REMOTE	Specifies the type of Execution Environment to start within the address space. REMOTE is the only valid value for this parameter. REMOTE specifies that the IMS MPR establishes a connection with a Native Execution Environment specified by the EENAME parameter to access the Data Object Broker and external server data.

Testing the Software

Starting the Service Gateway for IMS TM

Service Gateway for IMS TM runs in an MPR. An MPR can be executed as a batch job or a started task. To start an Execution Environment in an IMS MPR, complete the following steps:

1. Start the Native Execution Environments where the MPRs are to log in.
2. Specify by an HRNIN DD in the MPR invocation JCL any parameters to override the Service Gateway for IMS TM defaults.

Starting the IMS MPR

Start the IMS MPR that processes TIBCO Object Service Broker transactions by doing one of the following:

- Submit the MPR invocation JCL manually.
- Issue the IMS command to have the IMS Data Object Broker submit the invocation JCL. For example:

  ```
  /STA REG membername
  ```

Logging In

To log in to TIBCO Object Service Broker running in an IMS TM environment, do one of the following:

- Type the TIBCO Object Service Broker IMS transaction name from an IMS terminal.

  The default transaction name provided is **`OSB`**. To log in to TIBCO Object Service Broker with your own user ID, use the **`OSB`** transaction name and specify your session parameters. For example:

  ```
  OSB U=userid,P=password
  ```

  In this example, *userid* and *password* are the user ID and password to log in to TIBCO Object Service Broker.

- Use the **`/FOR IMS`** command to display the TIBCO Object Service Broker login panel. Specify one of the transaction names coded when the TIBCO Object Service Broker IMS transactions were defined.

  Refer to Define TIBCO Object Service Broker IMS Transactions on page 133 for more information on how to define TIBCO Object Service Broker transactions.

To log in to TIBCO Object Service Broker with the transaction code equal to the login user ID, use the trancode associated with PSB **S6BIMTC2**. For example, if the trancode is OSBLOGON, type:

```
OSBLOGON
```

In this example, OSBLOGON is the user ID that is logged in to TIBCO Object Service Broker.

Stopping the Service Gateway for IMS TM

When an MPR is shut down, the Service Gateway for IMS TM within the MPR is terminated automatically

Installing OTMA Support

IMS TM Open Transaction Manager Access (OTMA) allows access to IMS applications and is supported by TIBCO Object Service Broker. Use of OTMA is discussed in *TIBCO Object Service Broker for z/OS External Environments*; the underlying rule interface tools are described in *TIBCO Object Service Broker Shareable Tools.*

Refer to the IBM IMS documentation for configuration guidance; in particular, the *IMS V9 OTMA Guide and Reference* manual. This document also contains the return codes you may encounter when using the OTMA interface. Pay particular attention to the values specified for the IMS TM started task parameters:

`GRNAME=gggggggg`	XCF group IMS is to join
`OTMA=Y`	Enable OTMA
`OTMANM=mmmmmmmm`	XCF member name IMS uses for the group

Some of these values will have to be specified in calls from TIBCO Object Service Broker to OTMA.

To configure your IMS TM system to support OTMA, apply USERMODF in the CNTL library to link the OTMA interface stub DFSYCRET into S6BDRSES.

A sample rule, `SAMPLE_OTMA_CALL`, is provided to invoke the IMS sample verification transaction PART. You will need to modify this sample rule to be compatible with the IMS TM started task parameters mentioned above. Specifically, the values in statement 3 of the sample rule must be changed to match your IMS parameters, as follows:

```
CALL @OTMAOPEN('<GRNAME_IMS_parm_value>', MEMBER,
'<OTMANM_IMS_parm_value>')
```

The value of the `GRNAME` parameter should replace `<GRNAME_IMS_parm_value>` and the value of the `OTMANM` parameter should replace `<OTMANM_IMS_parm_value>`.

To verify the interface, run the IBM sample transaction PART under IMS TM. If successful, modify and execute the sample rule as shown above.

```
EX: SAMPLE_OTMA_CALL(USER01)
```

A successful OTMA call will provide the following output:

```
OTMA error Function=OPEN RC=0 reason1=0 reason2=0 reason3=0 reason4=0
OTMA error Function=ALLO RC=0 reason1=0 reason2=0 reason3=0 reason4=0
OTMA error Function=SEND RC=0 reason1=0 reason2=0 reason3=0 reason4=0

          Part...........     AN960C10; Desc........... WASHER
          Proc Code......           74; Inv Code.......        2
          Make Dept......        12-00; Plan Rev Num...
          Make Time......           63; Comm Code......       14

OTMA error Function=FREE RC=0 reason1=0 reason2=0 reason3=0 reason4=0
OTMA error Function=CLOS RC=0 reason1=0 reason2=0 reason3=0 reason4=0
```

Chapter 7 Installing the Service Gateway for WMQ

This chapter describes how to install the Service Gateway for WMQ.

Topics

Introduction

Service Gateway for WMQ is an interface used to access IBM WebSphere MQ message queues from within TIBCO Object Service Broker. It ensures that data is presented in a manner consistent with TIBCO Object Service Broker behavior.

Deployment

You can configure the Data Object Broker and Service Gateway for WMQ to reside on different hosts and/or operating systems (z/OS, Windows or Solaris).

See Also *TIBCO Object Service Broker for z/OS External Environments* or *TIBCO Object Service Broker for Open Systems External Environments* for more information on using Service Gateway for WMQ.

Preparing for Installation

You must install the TIBCO Object Service Broker base component before installing Service Gateway for WMQ. For installation on z/OS, you must also have the `<HLQ>.INSTALL` data set that was created during that installation.

Distribution Media and Contents

This section describes how to obtain the software, and the installation file that comprises the distribution media. Similar to the TIBCO Object Service Broker base component, the Service Gateway for WMQ software is distributed in `.xm1` format within a ZIP file.

Distribution File Format

The file is in a format compatible with IBM System Modification Program/Extended (SMP/E) naming conventions. The product is packaged in SMP/E txlib format.

Obtaining the Installation Media

You can download the software from the TIBCO Software web site by following these steps:

1. Contact TIBCO Software Inc. for a password, directory information, etc.
2. Connect to the TIBCO web site with the required information.
3. Download the appropriate zip file.

For details on installation on z/OS, see Uploading the Software on page 146, followed by Installing the Software on page 147.

Installation Files

The following zip file comprises the distribution media:

```
TIB_srvcgw-wmq_6.0.0_zos.zip
```

Uploading the Software

If you have acquired the Service Gateway for WMQ software by downloading it from the TIBCO Software web site, you must upload the software to the z/OS host system.

Preparing and Uploading the Product File

1. Download or copy the `TIB_srvcgw-wmq_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.
2. Unzip the file to a temporary location on the PC. The zip file contains multiple files; of these, the following file is the only file used in this installation:

   `wmq.xm1` – compressed file containing Service Gateway for WMQ
3. Pre-allocate the following sequential data set on the z/OS host system:

   `<HLQ>.WMQ.XM1 (size 10 KB)`

   Use the same <HLQ> that you specified when you uploaded the base component. Below is sample JCL to allocate this data set. Provide a JOB card and submit the JCL.

```
//ALLOC EXEC PGM=IEFBR14
//DD1   DD DSN=<HLQ>.WMQ.XM1,
//      DISP=(,CATLG,DELETE),UNIT=SYSDA,
//      DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
//      SPACE=(TRK,(2,1))
```

4. FTP the `wmq.xm1` file in BIN mode to the `<HLQ>.WMQ.XM1` data set.

Installing the Software

You must perform the installation under an ISPF environment.

This section describes the procedure for installing the Service Gateway for WMQ on z/OS. These instructions are for a new installation and for migrating from a previous version.

You can start the installation if you have the following data sets ready:

- <HLQ>.INSTALL
- <HLQ>.WMQ.XM1

You must use the <HLQ>.INSTALL data set that was created during the installation of the TIBCO Object Service Broker base component.

Checklist of system environment information you have to provide for this install:

- High level qualifier of the MQ Series load library

Edit the Properties File

Edit the PROPERTY member in <HLQ>.INSTALL. Table 7 describes the keywords in the properties file for installing this component.

Table 7 Properties File Keywords

Keyword	Description
INSTALL=	To install a local Service Gateway for WMQ, specify WMQ: INSTALL=WMQ
MQMLIB=	The high level qualifier of the MQ Series load library SCSQLOAD; for example, MQM.

If member PROPERTY was used for another install, save it by renaming the member, and then create a new PROPERTY member with the keywords above.

Initial Installation

STEP 1: **Execute File Tailoring EXEC to start installation.**

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

The installer verifies the value of the `MQMLIB keyword`. If valid, the installer creates the JCL to complete the pre-installation phase.

STEP 2: **Run Job WMQ.JCL.**

This batch job will uncompress the `WMQ.XM1` file to produce the distribution library.

JCL in: `<HLQ>.WMQ.JCL (Edit the JOB card to your site's standards)`

Data Set: `<HLQ>.WMQ.JCL (SUB data set)`

Uncompressing `<HLQ>.WMQ.XM1` produces the distribution library `<HLQ>.WMQ.FILEI`.

STEP 3: **Create and customize work copies of data sets.**

Member in: `<HLQ>.WMQ.FILEI`

Member: `S6J1CUST (EX member)`

The following work copies are created and customized with values specified by OSEMOD variables:

- `<HLQNONV>.<INSTVER>.JCL` – sample JCL
- `<HLQNONV>.<INSTVER>.WMQ.JOBS` – install jobs for WMQ

STEP 4: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.WMQ.JOBS`

Member: `S6J2RUNJ (EX member)`

SEND messages are directed to the userid specified in the NOTIFY parameter of each job submitted, informing the user of submission, and normal completion or abnormal termination. On successful completion of the final job in JOBSJ list, the message ALL MEMBERS PROCESSED displays. This completes the auto-installation process for WMQ.

STEP 5: **Perform APPLY of CICS Execution Environment (Optional).**

If there is a requirement to run an instance of the Gateway in a CICS Execution Environment, perform the following:

Member: `<HLQNONV>.<INSTVER>.WMQ.JOBS(S6J4APLY) (Edit the JOB card to your site's standards)`

Member: `S6J4APLY (SUB member)`

STEP 6: **Rework USERMODF (required if IMS TM was installed previously).**

Member in: `<HLQNONV>.<INSTVER>.JCL`

Member: `USERMODF`

Increment value of REWORK by 1, add REDO to the APPLY statement and re-submit the job.

Verification of Installation

Before running the IVP, you must know the names of a queue manager and a queue from which you can read and to which you can write.

From the workbench or in batch, do the following:

1. Run the rule `@MOMIVP_WRITE(queue_manager_name,queue_name')`.

   The process ends with this message:

   ```
   MOMIVP: MSG"IVP TEST MSG" WRITTEN TO QUEUE "queue_name"
   ```

2. Run the rule `@MOMIVP_READ(queue_manager_name,queue_name')`.

   The process ends with this message:

   ```
   MOMIVP: MSG "IVP TEST MSG" READ FROM QUEUE "queue_name"
   ```

Chapter 8 Testing the TIBCO Object Service Broker SDK (C/C++)

This chapter describes how to test the installation of the TIBCO Object Service Broker SDK for C and C++.

Topics

Introducing SDK (C/C++)

The SDK (C/C++) is an extension of the Call Level Interface. It extends the interface beyond the boundaries of the Execution Environment.

The SDK (C/C++) is included with the installation of the TIBCO Object Service Broker base component. At the completion of the install process, the following should be in place:

- The SDK (C/C++) header file for entry points declaration, OSCLI, located in $HLQNONV$.$INSTVER$.H
- The sidedeck file definition, OSCLI, located in the CNTL data set. It contains import code statements for all the available SDK (C/C++) functions

See Also *TIBCO Object Service Broker for z/OS External Environments* for more information about the SDK (C/C++) server and about using the SDK (C/C++).

Test the SDK (C/C++)

Start the SDK (C/C++) Server

Refer to *TIBCO Object Service Broker for z/OS External Environments* for information on Execution Environment considerations when running the SDK (C/C++) server.

Access the SDK (C/C++) Server

Refer to *TIBCO Object Service Broker for z/OS External Environments* for information on a sample program that you can use to test your SDK (C/C++) installation.

Chapter 9 Using Data Object Broker User Exits

This chapter describes how to customize your system with Data Object Broker user exits.

Topics

Overview

This chapter identifies and explains the purpose and use of the Data Object Broker user exits provided by TIBCO Object Service Broker. You are responsible for the development and support of the routines invoked at these user exits.

User Exits

The Initialization exit is the only one that you must supply, because it provides a road map to the others.This is done via USERMODB as described in Sample Template and Application on page 186

The following user exits are available:

User Exit	When Called	See Page
Cache Complete	After the last Cache buffer is built, before I/O is requested.	159
Checkpoint Start	Before Cache I/O is initiated.	160
Command	For a user command, which starts with a “Z”.	161
File Management Checkpoint	When the Cache is swapped for the next checkpoint.	163
Initialization	Before the start of parameter parsing.	158
Initialization Complete	At the end of phase 1 initialization.	164
Open	When initial file open processing completes.	165
Parameter	For a user parameter, which starts with a “Z”.	166
Recovery	For each redolog item reapplied.	167
Redolog-1	After the first I/O for an intent is issued.	168
Redolog-2	After the last I/O for an intent is completed.	170
Termination	When the Data Object Broker is terminating (normally or abnormally).	172

Usage Notes

When writing your exit routines, note the following:

- Routines must adhere to a predefined parameter list and set of return codes.
- User commands and parameters must start with the character “Z”.
- Other than data specifically passed to a user exit, no attempt to access or monitor Data Object Broker activity is permitted from a user exit.
- All processing in a user exit must be self-contained, that is, Data Object Broker services are not available to the exits.
- Control passes to the user exits using the Branch and Stack (BAKR) instruction. Before the exit is called, Register 13 is set to zero. The exit is APF-authorized and is entered in 31-bit addressing mode and in storage-key 8.
- Exits must return control to the Data Object Broker via the Program Return (PR) instruction.
- If the Data Object Broker is executing with ESTAE support (this is the default and is recommended), when an abend occurs in a user exit, the Data Object Broker generates a message in its log. All user exit abends, except those from the Command exit, terminate the Data Object Broker. The Command exit is called from OPERTASK and, when the exit abends, OPERTASK automatically restarts.
- The user exits are invoked under different Data Object Broker tasks. Refer to the Warning on page 156.

Register usage Notes

- Register 0 points to the general parameter list, which contains a pointer to shared storage and pointers to all active user exits.
- Register 1 points to an exit-specific parameter list.
- On return from an exit, register 15 is assumed to contain the exit completion code.

Using user exits is a powerful tool. Be very careful that you do not compromise your data or your TIBCO Object Service Broker system.

Serialization by TIBCO Object Service Broker Task

The exits are serialized by the task they are called from:

Task	Alias	User Exits Serialized
S6BCR000	SUPVTASK	Initialization, Parameter, Initialization Complete, Termination.
S6BCR600	OPERTASK	Command, Parameter.
S6BCR100	CHPTTASK	Checkpoint Start, Cache Complete.
S6BCR300	FILETASK	Open, Redolog-1, Redolog-2, File Management Checkpoint.
S6BCR500	REDOTASK	Recovery.

Take care when performing z/OS actions that are task-dependent. For example, a data set opened under the control of one task cannot be closed under the control of another task. A possible solution is to perform any such function within a subtask that you write and attach during Data Object Broker initialization and detach during termination.

For important information about serialization when enabling and disabling exits, refer to USREXTPL Usage and Mapping on page 173.

Processing Flow

During initialization the routine associated with the initialization exit is called from SUPVTASK. Assuming that this exit routine enables all other possible user exits, the processing proceeds as follows:

1. The Parameter exit routine is called for each user parameter specified in the input parameters under the control of SUPVTASK. Then the Initialization Complete exit routine is called under the control of SUPVTASK, the Open exit routine is called under the control of FILETASK, and the Recovery exit routine is called once for each recovered intent (if any).

   When the last recovered intent (if any) is passed to the Recovery exit, the Data Object Broker finishes its initialization process.

2. For every update intent processed by FILETASK, the Redolog-1 exit is called, followed by the Redolog-2 exit, for the same commit transaction.

   No other exit is called by FILETASK during this interval.

3. If a checkpoint is required, the File Management Checkpoint exit is called.
4. When a checkpoint is triggered, CHPTTASK calls the Checkpoint Start exit for this checkpoint, followed by the Cache Complete exit for the same checkpoint.

   CHPTTASK does not call any other exit during this interval.
5. For any given checkpoint, the File Management Checkpoint exit in FILETASK is called after the Checkpoint Start and Cache Complete exits in CHPTTASK for the previous checkpoint, and before the Checkpoint Start and Cache Complete exits for the current checkpoint.
6. If a user operator command is entered, the Command exit is called under control of OPERTASK.
7. Similarly, if a command of the following format is entered, the Parameter exit is called under the control of OPERTASK:

   ```
   F DOBNAME,PARAMETER=Zuserparm=value
   ```
8. When the Data Object Broker is terminated, the Termination exit is called under the control of SUPVTASK.

User Exits

Initialization Exit

An Initialization exit called USRX0000 must be present, to set the pointers for all user exits that are to be called.The Initialization exit does this by turning off the high-order bit for the address for that user exit.

If USRX0000 is present in the load library and the Data Object Broker Parameter USEREXIT is set to Y, then the exits are initialized.

The Data Object Broker terminates with an error message if USRX0000 is present in the load library and the Data Object Broker Parameter USEREXIT is set to N, or USRX0000 is not present in the load library and the Data Object Broker Parameter USEREXIT is set to Y.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	=0 – no user specific parameter list.
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Initialization exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker initialization continues.
<>0	Error	Data Object Broker terminates with Abend U049.

Cache Complete Exit

The Cache Complete exit is called after the last Cache buffer has been built and before the write request is issued. You can use this exit to ensure that all intent list user processing is complete. Invoking the exit routine before the last Cache buffer is written ensures that the transactions for the checkpoint are recovered in the event of an abnormal termination of the Data Object Broker.

For important information about event timing, refer to the Warning on page 160.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the exit parameter list where: +00(4) – checkpoint number
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Cache exit routine entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
> 0	Error	Data Object Broker abends with code U107.

Checkpoint Start Exit

Pending page image updates are written serially to one of the Cache data sets before the asynchronous processing of the page image data sets and journals. Before the Cache write is initiated, the Checkpoint Start exit is invoked.

If the custom processing collects intents during redolog processing with the intention of processing the intent lists at checkpoint time, this is the time to initiate the processing.

There is a window for a possible timing error if intents are not done their required processing before the last Cache buffer is processed. If a Data Object Broker termination occurs during this window, user exit processing is passed some intents twice. You must ensure that recovery intents are processed once and only once.

This happens because, during restart recovery, each recovered intent list is presented to the Recovery exit. If the Checkpoint Start or Cache Complete exit recorded any captured intents and the Data Object Broker terminates before the "last Cache" I/O is complete, upon restart these intents are passed to the Recovery exit for processing.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the exit parameter list where: +00(4) – checkpoint number
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of the exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
> 0	Error	Data Object Broker abends with code U107.

Command Exit

If your application requires dynamic external influence on the processing, the exits must support user operator commands. To support user commands, the optional Command exit must be enabled. If there are no user commands, the Command user exit should remain inactivated (that is, USREXTPL.UXCMND@ = 0). If present, the Command exit is called when a command received from the operator starts with the character “Z”. The exit is responsible for producing any log notification that results from user commands, such as return code 0 or 4.

A user command must be 5 to 14 alphanumeric characters long starting with a “Z”. The format of the command should be the same as for other keyword Data Object Broker operator commands with a value, such as “ZUSER=value”, or without a value, such as “ZSPIN”.

For important information about serialization when enabling and disabling exits, refer to USREXTPL Usage and Mapping on page 173.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to Command exit parameter list where: +00(4) – keyword length (relative to zero) +04(4) – keyword start address +08(4) – value length (relative to zero, undefined if no value supplied) +12(4) – value start address or 0 if no value supplied
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Command exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
4	Invalid user command	The command is ignored and Data Object Broker processing continues.
> 4	Unrecognized command	The command is ignored and Data Object Broker processing continues.

File Management Checkpoint Exit

A checkpoint can be triggered based on volume of processing, a timed event, an operator command, or via the user Redolog-1 and Redolog-2 exits. When preparing for a checkpoint, the File Management routine switches the Cache buffer. When this Cache switching is about to occur, the File Management Checkpoint user exit is invoked.

If the user exits are collecting intents within a checkpoint boundary using alternate buffers, this exit would be the appropriate place to switch to the alternate buffer.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the exit parameter list where: +00(4) – checkpoint number
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of the exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
> 0	Error	Data Object Broker abend U320.

Initialization Complete Exit

After completing phase-1 initialization, the Data Object Broker calls the Initialization Complete exit. During Phase 1, the Data Object Broker obtains the majority of the space it requires, allocates the control data sets, and makes any parameter adjustments necessary, such as the values of checkpoint, transaction, and page limits. The Initialization Complete exit provides the best opportunity for the calculation and acquisition of user storage. The UXSPACE@ field in the USREXTPL is available to store the start address of the user storage block. This is also the time to perform sanity checks on the user parameters that are processed.

If properly managed, it is more efficient to obtain all the required user storage in a single block during initialization rather than repeatedly obtaining and releasing storage during the lifetime of the Data Object Broker.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the exit parameter list where: +00(4) – pointer to the Data Object Broker Parameter Setting Parse Data block (refer to Data Object Broker Initialization Done Data Block on page 174).
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of the exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker initialization continues.
> 0	Error	Initialization fails with abend U049.

Open Exit

The Open exit is called at the end of the File Management third-stage initialization. At this point, the TIBCO Object Service Broker data sets required to start processing are allocated and opened. The Open exit provides the opportunity to open the data sets you require.

For important information about opening and closing data sets, refer to the Warning on page 156.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	= zero.
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Open exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
> 0	Error	Data Object Broker initialization fails with a U301 abend.

Parameter Exit

If your application requires external influence, you can accommodate this with user runtime parameters. To support user parameters, the optional Parameter exit must be enabled. If there are no user parameters, the Parameter user exit should be left inactivated (that is, USREXTPL.UXPARM@ = 0). The Data Object Broker invokes parameter processing during initialization and when a "PARAMETER=" operator command is received. When the Data Object Broker encounters a parameter keyword starting with the character "Z", it invokes the user parameter exit if UXPARM@ is a positive non-zero value. The exit routine is responsible for doing any user-parameter reporting required.

A user parameter must be 5 to 16 alphanumeric characters long starting with a "Z". The format of the parameter should be the same as for other Data Object Broker runtime parameters with a value, for example, "ZUSER=value".

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to Parameter exit parameter list where: +00(4) – keyword length (relative to zero) +04(4) – keyword start address +08(4) – value length (relative to zero) +12(4) – value start address
2-12	Unpredictable.

Calling Register	Contents
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Parameter exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker initialization continues.
4	Invalid user parameter	If the exit routine is called during Data Object Broker initialization, initialization fails with abend U049. If the exit routine is called as the result of a PARAMETER= command, the parameter is ignored.
> 4	Unrecognized Parameter	If the exit routine is called during Data Object Broker initialization, initialization fails with abend U049. If the exit routine is called as the result of a PARAMETER= command, the parameter is ignored and Data Object Broker processing continues.

Recovery Exit

During Data Object Broker start-up processing after an abnormal termination of the Data Object Broker, the intent lists saved on the redolog that have not reached checkpoint are reapplied. After each “recovery” intent has been processed, control passes to the Recovery exit.

Refer to Redolog Entries on page 175 for details of the layout of the redolog entries.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to Recovery exit parameter list where: +00(4) – pointer to intent list to recover +04(4) – length of intent list +08(4) – pointer to the redolog header +12(4) – current checkpoint number
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Recovery exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker initialization continues.
<>0	Error	Data Object Broker terminates with abend U049.

Redolog-1 Exit

After an intent list has been validated by the Data Object Broker, it is written to the redolog. TIBCO Object Service Broker uses an asynchronous write so that I/O processing of the primary and duplex redologs can overlap. TIBCO Object Service Broker calls the Redolog-1 exit after issuing the asynchronous write request.

When the intent is written to the redolog, it could be split into segments depending on the space available in the VSAM Control Interval. In this case, TIBCO Object Service Broker calls the Redolog-1 exit after issuing the asynchronous write request to the redolog for the first block.

The Redolog-1 and Redolog-2 exits are designed to work in series to provide a processing window that overlaps the I/O processing of the redolog. Processing requirements in Redolog-2 are dependent on the processing initiated in Redolog-1.

Refer to Redolog Entries on page 175 for details of the layout of the redolog entries.

If the Redolog-1 exit is performing selection on the intent list and the current intent is not selected, there can be performance advantages if the Redolog-2 exit is bypassed. To disable the Redolog-2 exit, turn on the high-order bit for its pointer in the general parameter list (UXREDO2@).

The processing at this point is on the TIBCO Object Service Broker critical path. Any additional processing causes degradation of commit turnaround time. Make sure your application takes this into account.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the Redolog-1 exit parameter list where +00(4) – Address of the intent list +04(4) – Length of the intent list being processed +08(4) – Pointer to the redolog header +12(4) – The current checkpoint number +16(8) – User ID of the transaction owner +24(4) – Unique connection pointer of the transaction owner
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Redolog-1 exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
4	Force checkpoint	Processing completed successfully and you must take a checkpoint as soon as possible.
> 4	Error	Data Object Broker abends with code U320.

Redolog-2 Exit

As stated under Redolog-1 Exit on page 168, the write request of the intent to the redolog is asynchronous. After the check to verify a successful write was satisfied, the Redolog-2 exit is called. If the intent is written over a number of redolog VSAM Control Intervals, TIBCO Object Service Broker calls the Redolog-2 exit only after successfully writing the last Control Interval.

The Redolog-1 and Redolog-2 exits are designed to work in series to provide a processing window that overlaps the I/O processing of the redolog. Processing requirements in Redolog-2 are dependent on the processing initiated in Redolog-1.

Refer to Redolog Entries on page 175 for details of the layout of the redolog entries.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to Redolog-2 exit parameter list where +00(4) – Address of the intent list +04(4) – Length of the intent list being processed +08(4) – Pointer to the redolog header +12(4) – The current checkpoint number +16(8) – User ID of the transaction owner +24(4) – Unique connection pointer of the transaction owner
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Redolog-2 exit entry point.

Return Codes

Upon return from the exit routine the contents of Register 15 determine further processing as follows:

Return Code	Meaning	Action
0	Success	Data Object Broker processing continues.
4	Force checkpoint	Processing completed successfully and you must take a checkpoint as soon as possible.
> 4	Error	Data Object Broker abends with code U320.

Termination Exit

The Termination exit is called when the File Management and Checkpoint functions are done.

If the user exits have a data set open, this exit provides a point to close them cleanly. For important information about opening and closing data sets, refer to the Warning on page 156.

Registers

Calling Register	Contents
0	General Parameter List (refer to USREXTPL Usage and Mapping on page 173).
1	Pointer to the exit parameter list where: +00(4) – 0 if Data Object Broker processing a normal termination
2 – 12	Unpredictable.
13	=0 – no register save area is supplied.
14	Next executable instruction (return point).
15	Address of Termination exit entry point.

Return Codes

The return code for the Termination exit is ignored.

Data Layouts

USREXTPL Usage and Mapping

In USREXTPL, if a pointer (except for the one for the Initialization exit) is 0 or has the high-order bit turned on, the corresponding exit routine is not invoked during Data Object Broker processing. You turn the high-order bit off to enable the exit. You can use this to dynamically control which exits are taken. If you use this technique, you must take care to ensure correct serialization of the user exits.

This is a map of USREXTPL, the general parameter list of the user exits (all fields default to zero):

Offset	Length	Field	Description
0	4	UXSPACE@	Pointer to shared storage acquired by Initialization exit.
4	4	UXINITX@	Pointer to the Initialization user exit.
8	4	UXPARM@	Pointer to the Parameter user exit.
12	4	UXIDON@	Pointer to the Initialization Complete user exit.
16	4	UXCMND@	Pointer to the Command user exit.
20	4	UXOPEN@	Pointer to the Open user exit.
24	4	UXRCVRY@	Pointer to the Recovery user exit.
28	4	UXREDO1@	Pointer to the Redolog-1 user exit.
32	4	UXREDO2@	Pointer to the Redolog-2 user exit.
36	4	UXFCHPT@	Pointer to the File Management Checkpoint user exit.
40	4	UXCHPTS@	Pointer to the Checkpoint Start user exit.
44	4	UXCACHE@	Pointer to the Cache Complete user exit.
48	4	UXTERM@	Pointer to the Termination user exit.
52	4	UXRSRV1	Reserved for future growth.

Offset	Length	Field	Description
56	4	UXRSRV2	Reserved for future growth.
60	4	UXRSRV3	Reserved for future growth.

Data Object Broker Initialization Done Data Block

This is a map of the Data Object Broker Parameter Setting data block:

Offset	Length	Field	Description
0	16	IDNODENM	16-character Node Name.
16	8	IDSTART	8-character date in YYYYMMDD format.
24	8	IDCOMMID	8-character communications identifier.
32	2	IDMAXU	Half-word maximum user count.
34	2	IDMAXT	Half-word maximum number of concurrent messages being processed.
36	2	IDCHPAGE	Half-word maximum number of pages per checkpoint.
38	2	IDCHTRX	Half-word maximum number of transactions per checkpoint.
40	2	IDRESPAG	Half-word number of resident pages requested.
42	2	IDNETVW	Half-word Netview alert base.
44	1	IDGTFID	One-byte GTF record identifier.
45	1	IDSMFREC	One-byte SMF record number.
46	8	IDSNAPP	Remote printer JES user ID for snap dumps.
54	1	IDSNAPC	Snap dump class.
55	73	IDFILLER	Space for additional parameters.

Redolog Entries

The redolog has two distinct formats. To determine the applicable format, look at the first two bytes.

Format 1:

If the first two bytes of the intent list are X'0001', the intent list is in format 1, which is described in the following tables.

Format 1 Header Layout:

Offset	Length	Field	Description
+00	2	R1H#TRX	Number of transaction in the intent. Always X'0001'.
+02	1	R1HLOCK	Logical lock action: • " " (X'40') retain locks • "R" (X'D9') release locks (end of transaction)
+03	4	R1HTRXID	Transaction identifier.
+07	2	R1H#UPD	Number of updates within the intent.

Format 1 Intent Body Layout:

Offset	Length	Field	Description
+00	2	R1BLEN	Length of the update segment (does not include itself).
+02	1	R1BREQ	Processing requested: "D" – delete occurrence "I" – insert occurrence "R" – replace occurrence
+03	1	R1BTABL	Table name length.
+04	V	R1BTABNM	Variable-length table name.
		R1PARMS [a]	Parameter values, if required.
		R1BROWL [b]	Length of occurrence (does include itself).

Offset	Length	Field	Description
		R1BF1L [c]	Length of first data field. Length can be 1 or 2 bytes. If the high-order bit is on, it is a 2-byte length.
		R1BF1V [c]	Variable-length context on field 1.

a. R1PARMS contains the table parameter values in variable format if they are required for the table being updated. The entries consist of a 1-byte length field (the maximum length of an individual parameter is 127) followed by the data for each parameter.

b. R1BROWL is the length of the data occurrence.

c. R1BF1L and R1BF1V are repeated for each field in the table. If the high-order bit of R1BF1L is on for a field, this length field is two bytes long and the actual data length can be obtained by turning off this high order bit.

Format 2

If the first two bytes of the intent list are not X'0001', the intent list is in format 2, which is described in the following tables.

Format 2 Header Layout:

Offset	Length	Field	Description
+00	2	R2HLEN	Length of the Intent-list data (includes itself, is 2 bytes shorter than the length passed as the second parameter to the Redolog-1, Redolog-2, and Recovery exits).
+02	2	R2H#TRX	Number of transactions in the intent list.
+04	4	R2HTRXID	Transaction identifier.
+08	1	R2HLOCK	Logical lock action: " " (X'40') retain locks "R" (X'D9') release locks (end of transaction)
+09	1		Not used.
+10	2	R2H#UPD	Number of updates within the intent.

Format 2 Intent Body Layout:

Offset	Length	Field	Description
+00	2	R2BLEN	Length of the update segment (ignore the high-order bit).
+02	1	R2BREQ	Processing requested: “E” – delete secondary index “K” – clear table “S” – build secondary index “@” – table relocation
+03	1	R2BRSV1	Reserved byte.
+04	4	R2BTRXID	Transaction identifier.
+08	16	R2BTABNM	Table name.
+24	3	R2BTBTYP	Table type.
+27	1	R2BLOCK	Logical lock action (not currently used): “ ” (X'40') retain locks “R” (X'D9') release locks (end of transaction)
+28	2	R2BOPARM [a]	Offset to parameter instance block from start of body.
+28	2	R2BSEG#	Segment number for MOVTAB processing.
+30	26	R2BORSV2	Reserved.
+56	2	R2BOMISC [b]	Offset to key field for secondary index build or delete.
+58	6	R2BORSV2	Reserved.

a. R2BOPARM, if non zero, contains an offset from the start of this Intent body to the parameter data for this request. The parameter values are in variable format if they are present for the table being processed. The entries consist of a 1-byte length field (the maximum length of an individual parameter is 127) followed by the data for each of the parameters. For a MOVTAB request, this field is overlaid with the segment number to which the table is to be relocated.

b. R2BOMISC, if non-zero, contains an offset from the start of this Intent body to the name of the key to be used for a Secondary Index Build or Delete. The Key value is 16 bytes long with the key padded with blanks on the right if necessary.

Redolog Header

Offset	Length	Field	Description
+00	2	RLENTLEN	Length of redolog entry.
+02	1	RLTYPE	Entry type: X'01' – FILETASK initialized X'02' – Normal shutdown X'04' – Segment online X'08' – Segment offline X'10' – FILETASK notified, checkpoint complete X'20' – Intent list entry X'40' – Last block of chain [a] X'80' – Checkpoint request initiated X'00' – Dummy entry
+03	1	RLBLOCK# [a]	Segment sequence number.
+04	4	RLCCHPT#	Current checkpoint number.
+08	4	RLLCHPT#	Last completed checkpoint number.
+12	4	RLJRNRBA	Current journal RBA.
+16	1	RLJRNLID	Current journal ID.
+17	1	RLJRNSPN	Journal spin status.
+18	2	RLDATAL	Total data length.
+20	8	RLTIME	TOD time stamp (generated by a STCK).
+28	4	RLSEQ# [b]	COMMIT Transaction sequence number or 0 (zero).
+32	4	RLRRBA	Redolog RBA.
+36	4		Reserved.
+40	4	RLEYEC1	Eye Catcher: “REDO”.
+44	4	RLEYEC2	Eye catcher type for data intents: “DATA”.

a. The RLBLOCK# and the x'40' bit in the RLTYPE field can be ignored. These are used internally by TIBCO Object Service Broker when it splits intents across multiple 4-KB records in its redolog data set. Only Intent-list entries are seen by the user-exit processing. The field RLTYPE for these entries is either x'20' or x'60'. The Redolog-1 and Redolog-2 exits are passed the complete Intent-list entry once for each commit request.

b. For an Intent-list entry, the field RLSEQ# is a number between x'00000001' and x'00FFFFFE' that is incremented for each Commit transaction processed by the Data Object Broker. For Intents passed to the Redolog-1 and Redolog-2 exits, this number is incremented sequentially and wraps back to 1 when the maximum possible value is reached. Under certain circumstances, it is possible for this value to be incremented by more than 1 for consecutive intents passed to the Redolog-1 and Redolog-2 exit processing.

Redolog-Entry Samples

Sample Table Definition

These samples use the EXIT_SAMPLE table, which is defined as follows:

```
COMMAND==>                          TABLE DEFINITION

      Table: EXIT_SAMPLE      Type: TDS   Unit: USR040             IDgen: N
     Source:
   Parameter Name  Typ Syn Len Dec Class             '      Event Rule   Typ Acc
   ----------------  -   -- --- --    -              '   ---------------- -   -
_ PARAMETER1         S   C    4  0    D              ' _
_ PARAMETER2         S   C    4  0    D              ' _
_ PARAMETER3         S   C    4  0    D              ' _
_ PARAMETER4         S   C    4  0    D              ' _
     Field Name    Typ Syn Len Dec Key Ord Rqd    Default          Reference
   ---------------- -   -- ---- --  -   -   - ---------------- ----------------
_ KEY               Q   B     4  0  P
_ DATA              S   V   256  0
_
_
_
_
_
_
_
_
PFKEYS: 3=END 12=CANCEL 22=DELETE 13=PRINT 14=FIELDS 21=DATA 2=DOC
```

The following samples are produced on entry to the Redolog-1 user exit. For the SIXBUILD and SIXDELETE samples, the length of field DATA is reduced to 127.

Sample 1

This is a format-1 intent list produced by adding two rows to the EXIT_SAMPLE table for the 'P1','P2','P3','P4' instance with keys of 1 and 2. The data field of the second row is padded with the character “X” to illustrate a two-byte length field.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                      46E48C20 00000176          *.U......*
0007A2B0    0007A0B0 00000018 C4E9C3F1 F0404040  *........DZC10   *
0007A2C0    00000001                             *................*
```

Redolog header passed to the Redolog-1 for this intent:

```
0007A0B0    01A66001 00000018 00000018 00063000  *.w-............*
0007A0C0    01000176 BC1A974D C5F0AC00 000000E5  *......p(E0.....V*
0007A0D0    0000C139 0000E020 D9C5C4D6 C4C1E3C1  *..A...\.REDODATA*
```

Actual format-1 Intent entry:

```
46E48C20  0001D900  000E2C00  02004AC9  0BC5E7C9  *..R.......¢I.EXI*
46E48C30  E36DE2C1  D4D7D3C5  02D7F102  D7F202D7  *T_SAMPLE.P1.P2.P
46E48C40  F302D7F4  00310101  2CE3C8C9  E240C9E2  *3.P4.....THIS IS*
46E48C50  40E3C8C5  40C6C9D9  E2E340D9  D6E640D6  *THE FIRST ROW O*
46E48C60  C640C4C1  E3C140C9  D540E3C8  C9E240E3  *F DATA IN THIS T*
46E48C70  C1C240D3  C5011FC9  0BC5E7C9  E36DE2C1  *AB LE..I.EXIT_SA*
46E48C80  D4D7D3C5  02D7F102  D7F202D7  F302D7F4  *MPLE.P1.P2.P3.P4*
46E48C90  01060102  8100E3C8  C9E240C9  E240E3C8  *....a.THIS IS TH*
46E48CA0  C540E2C5  C3D6D5C4  40D9D6E6  40D6C640  *E SECOND ROW OF *
46E48CB0  C4C1E3C1  40C9D540  E3C8C9E2  40E3C1C2  *DATA IN THIS TAB*
46E48CC0  D3C54B40  C9E340C9  E240D7C1  C4C4C5C4  *LE. IT IS PADDED*
46E48CD0  40E6C9E3  C840E3C8  C540C3C8  C1D9C1C3  * WITH THE CHARAC*
46E48CE0  E3C5D940  E740E3D6  40E2C8D6  E640C140  *TER X TO SHOW A *
46E48CE0  E3C5D940  E740E3D6  40E2C8D6  E640C140  *TER X TO SHOW A *
46E48CF0  E3E6D640  C2E8E3C5  40D3C5D5  C7E3C840  *TWO BYTE LENGTH *
46E48D00  C6C9C5D3  C44BE7E7  E7E7E7E7  E7E7E7E7  *FIELD.XXXXXXXXXX*
46E48D10  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D20  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D30  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D40  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D50  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D60  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D70  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D80  E7E7E7E7  E7E7E7E7  E7E7E7E7  E7E7E7E7  *XXXXXXXXXXXXXXXX*
46E48D90  E7E7E7E7  E7E7                        *XXXXXX..........*
```

Sample 2

This is a format-1 intent list produced by deleting the row of key 1 of the two rows added by the previous sample and replacing the row of key 2. The data field of the second row is padded with the character “Z” to illustrate a two-byte length field.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                    46E47420  00000176  *.U......*
0007A2B0  0007A0B0  00000018  C4E9C3F1  F0404040  *........DZC10   *
0007A2C0  00000001                              *................*
```

Redolog header passed to the Redolog-1 for this intent:

```
0007A0B0    01A66001 00000018 00000018 00063000  *.w-............*
0007A0C0    01000176 BC1A9E35 1497A900 000000F8  *.........pz....8*
0007A0D0    0000C139 000177E1 D9C5C4D6 C4C1E3C1  *..A.... REDODATA*
```

Actual format-1 Intent entry:

```
46E47420  0001D900  00161000  02004AC4  0BC5E7C9  *..R.......¢D.EXI*
46E47430  E36DE2C1  D4D7D3C5  02D7F102  D7F202D7  *T_SAMPLE.P1.P2.P*
46E47440  F302D7F4  00310101  2CE3C8C9  E240C9E2  *3.P4.....THIS IS*
46E47450  40E3C8C5  40C6C9D9  E2E340D9  D6E640D6  *THE FIRST ROW O*
46E47460  C640C4C1  E3C140C9  D540E3C8  C9E240E3  *F DATA IN THIS T*
46E47470  C1C240D3  C5011FD9  0BC5E7C9  E36DE2C1  *AB LE..R.EXIT_SA*
46E47480  D4D7D3C5  02D7F102  D7F202D7  F302D7F4  *MPLE.P1.P2.P3.P4*
46E47490  01060102  8100E3C8  C9E240C9  E240E3C8  *....a.THIS IS TH*
46E474A0  C540C4C1  E3C140C6  D6D940E3  C8C540D9  *E DATA FOR THE R*
46E474B0  C5D7D3C1  C3C5D4C5  D5E340D6  C640D9D6  *EPLACEMENT OF RO*
46E474C0  E640F24B  40C9E340  C9E240D7  C1C4C4C5  *W 2. IT IS PADDE*
46E474D0  C440E6C9  E3C840E3  C8C540C3  C8C1D9C1  *D WITH THE CHARA*
46E474E0  C3E3C5D9  40E940E3  D640E2C8  D6E640C1  *CTER Z TO SHOW A*
46E474F0  40E3E6D6  40C2E8E3  C540D3C5  D5C7E3C8  *TWO BYTE LENGTH*
46E47500  40C6C9C5  D3C44BE9  E9E9E9E9  E9E9E9E9  *FIELD.ZZZZZZZZZ*
46E47510  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47520  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47530  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47540  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47550  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47560  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47570  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47580  E9E9E9E9  E9E9E9E9  E9E9E9E9  E9E9E9E9  *ZZZZZZZZZZZZZZZZ*
46E47590  E9E9E9E9  E9E9                          *ZZZZZZ..........*
```

Sample 3

This is a format-2 intent list using $CLRTAB to clear the 'P1','P2','P3','P4' instance of EXIT_SAMPLE.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                      46E46F20 0000005A          *.U?....!*
0007A2B0    0007A0B0 00000018 C4E9C3F1 F0404040  *........DZC10   *
0007A2C0    00000001                             *................*
```

Redolog header passed to the Redolog-1 for this intent:

```
0007A0B0    008A6001 00000018 00000018 00063000  *..-.............*
0007A0C0    0100005A BC1A983D 81150700 000000E6  *...!..q.a......W*
0007A0D0    0000C139 0000E1C6 D9C5C4D6 C4C1E3C1  *..A... FREDODATA*
```

Actual format-2 Intent entry:

```
46E46F20  00580001  00000E3B  40000001  804ED200  *........ ....+K.*
46E46F30  00000E3B  C5E7C9E3  6DE2C1D4  D7D3C540  *....EXIT_SAMPLE *
46E46F40  40404040  E3C4E240  00400000  00000000  *    TDS . ......*
46E46F50  00000000  00000000  00000000  00000000  *................*
46E46F60  00000000  00000000  00000000  02D7F102  *............P1.*
46E46F70  D7F202D7  F302D7F4  B8000200  00000148  *P2.P3.P4........*
```

Sample 4

This is a format-2 intent list using SIXBUILD to build a secondary index on field DATA of EXIT_SAMPLE.

The definition length of field DATA is reduced from 256 to 127 for this sample so that the SIXBUILD can proceed.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                      46E48C20 00000064          *.U......*
0007A2B0    0007A0B0 00000018 C4E9C3F1 F0404040  *........DZC10   *
0007A2C0    00000001                             *................*
```

Redolog Header passed to the Redolog-1 for this intent:

```
0007A0B0    00946001 00000018 00000018 00063000  *.m-............*
0007A0C0    01000064 BC1AA069 59079200 0000011B  *.........k.....*
0007A0D0    0000C139 00019D4E D9C5C4D6 C4C1E3C1  *..A....+REDODATA*
```

Actual format-2 Intent entry:

```
46E48C20  00620001  00001D0C  40000001  804ED200  *........ .....S.*
46E48C30  00001D0C  C5E7C9E3  6DE2C1D4  D7D3C540  *....EXIT_SAMPLE *
46E48C40  40404040  00000040  00000000  00000000  *    ... ........*
46E48C50  00000000  00000000  00000000  00000000  *................*
46E48C60  00000000  00000000  00000000  C4C1E3C1  *..... ......DATA*
46E48C70  40404040  40404040  40404040  00000000  *            ....*
46E48C80  0000      |         |         |         |*................*
```

Sample 5

This is a format-2 intent list using SIXDELETE to delete a secondary index on field DATA of EXIT_SAMPLE.

The definition length of field DATA is reduced from 256 to 127 for this example so that the SIXDELETE can proceed.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                      46E47020 00000064          *.U......*
0007A2B0    0007A0B0 00000018 C4E9C3F1 F0404040  *........DZC10   *
0007A2C0    00000001                            *................*
```

Redolog header passed to the Redolog-1 for this intent:

```
0007A0B0    00946001 00000018 00000018 00063000  *.m-............*
0007A0C0    01000064 BC1AA387 37648F00 0000011C  *......tg........*
0007A0D0    0000C139 00019DE2 D9C5C4D6 C4C1E3C1  *..A....SREDODATA*
```

Actual format-2 Intent entry:

```
46E47020  00620001  00001D1A  40000001  8058C500  *........ .....E.*
46E47030  00001D1A  C5E7C9E3  6DE2C1D4  D7D3C540  *....EXIT_SAMPLE *
```

```
46E47040  40404040  00000040  00000000  00000000  *    ... ........*
46E47050  00000000  00000000  00000000  00000000  *................*
46E47060  00000000  00000000  00000000  C4C1E3C1  *..... ......DATA*
46E47070  40404040  40404040  40404040  00000000  *            ....*
46E47080  0000      |         |         |         |*................*
```

Sample 6

This is a format-2 intent list using MOVTAB to move table EXIT_SAMPLE to segment 2 from segment 1.

Parameter list passed to the Redolog-1 user exit:

```
0007A2A8                      46E46A20 0000004C          *.U¦....<*
0007A2B0    0007A0B0 00000018 C4E9C3F1 F0404040  *........DZC10   *
0007A2C0    00000001                             *................*
```

Redolog header passed to the Redolog-1 for this intent:

```
0007A0B0  007C6001  00000018  00000018  00063000  *.@-............*
0007A0C0  0100004C  BC1AA4A0  2872FB00  00000121  *...<..u.........*
0007A0D0  0000C139  0001A234  D9C5C4D6  C4C1E3C1  *..A...s.REDODATA*
```

Actual format-2 Intent entry:

```
46E46A20    004A0001 00001E40 40000001 80407C00  *.¢.....  .... @.*
46E46A30    00001E40 C5E7C9E3 6DE2C1D4 D7D3C540  *... EXIT_SAMPLE *
46E46A40    40404040 E3C4E240 00020000 00000000  *    TDS .........*
46E46A50    00000000 00000000 00000000 00000000  *................*
46E46A60    00000000 00000000 0000               *..............C.*
```

Sample Template and Application

Template

The USRX0001 member in the ASM data set contains a template for creating Data Object Broker user exits.

Application

The USRX0000 member in the ASM data set contains a sample application illustrating the use of the user exits feature. The USERMODB member in the JCL data set contains JCL to install this sample application. The sample application implements all the user exits with the exception of the REDOLOG-2 Exit.

The application captures all the intent lists containing updates to a specified table and outputs them to a print file.

This is the complete intent list containing a reference to the specified table, not just the intent-list entry for that table.

This sample implements four user parameters as follows:

ZCLASS	The SYSOUT class for the output data set. Default: A.
ZDDNAME	The first four characters of the DDNAME to be used for the output data sets. No default; if ZDDNAME is not present, the exits are disabled.
ZSTORE	The maximum number of intent-list entries matching the selection criteria before a checkpoint is forced. Must be a number between 5 and 9999. Default: 100.
ZTABLE	The name of the table to be used for selection criteria. Must be a valid table name. Default: spaces, so no intents are captured. Modification of these values except for the table name is not permitted after initialization. Use of the MODIFY DOBNAME,PARM=userparm=value is not permitted.

The sample implements two user commands as follows:

ZSPIN	After the next intent list is passed through the REDOLOG-1 user exit, a checkpoint is forced, the current output data set is closed, and a new one is allocated and opened by the checkpoint exits.
ZTABLE	After the next intent list is passed through the REDOLOG-1 user exit, the name of the table to be monitored is changed, a checkpoint is forced, the current output data set is closed, and a new one is allocated and opened by the checkpoint exits. If no value is specified, the table name is reset to spaces. The DDNAME used for the output data set is XXXXnnnn where XXXX is the value specified by the ZDDNAME parameter and nnnn is a number starting at 0001 and incremented by 1 each time a data set is closed and reallocated.

Sample Exit Brief Descriptions

Initialization Exit

The Initialization exit (USRX0000) obtains a working storage area (mapped by WORKSPAC DSECT) for use by all the other supplied exits, initializes it, and stores its address in field UXSPACE@ of the User Exit Parameter list. It then sets up the addresses of all the other supplied user exits in the User Exit Parameter list, sets default values in the working storage area and attaches a subtask to perform the OPEN, CLOSE, and OFFLOAD functions (refer to SYSOTCB Subtask on page 189).

Cache Complete Exit

The Cache Complete exit checks to see if an offload of captured intents is in progress. If it is, Cache Complete waits for the SYSOTCB subtask to signal that it is complete. Then, if a spin of the output data set is required, it posts the SYSOTCB subtask to close the current output data set, and then it posts the SYSOTCB subtask again to allocate and open a new output data set.

Checkpoint Start Exit

The Checkpoint Start exit checks to see if any captured intents are to be off-loaded. If they are, it posts the SYSOTCB task to start the offload process and sets a flag to indicate to the Cache Complete exit that an offload is in progress.

Command Exit

The Command exit implements the ZSPIN and ZTABLE commands as detailed in Template on page 186.

File Management Checkpoint Exit

The File Management Checkpoint exit swaps the intent-list collection buffers, if required, and sets pointers for the checkpoint user exits to offload captured intent lists. Checks are made to see if any activity such as a spin is required and if so, sets flags to control the checkpoint user exits accordingly.

Initialization Complete Exit

This exit checks that the value supplied for ZSTORE does not exceed the number of transactions per checkpoint. If it does, it is reset to the number of transactions per checkpoint. Storage is obtained for two buffers capable of holding intents for the number of intents specified by ZSTORE and various pointers are initialized in the common working storage.

Open Exit

The Open user exit checks to see if there is a value for ZDDNAME. If none is supplied, all the exits are disabled by turning on the high-order bits in their addresses in the User Exit Parameter list. If a value is supplied, the SYSOTCB subtask is posted to open the first output data set.

Parameter Exit

The Parameter user exit implements the user parameters described in Template on page 186.

Recovery Exit

The Recovery user exit captures any intents matching the selection criteria in one of the buffers obtained by the Initialization Complete exit and marks them as recovered intents. A check is made to stop these intents from overflowing the available storage. This should happen only if the parameter values were changed since the Data Object Broker shut down abnormally (for a normal shut down no intents would be recovered) or if monitoring is being done on initialization of a different table name than at the time of the failure.

Redolog-1 Exit

The REDOLOG-1 user exits captures intents matching the selection criteria in one of the buffers obtained by the Initialization Complete exit and marks them as normal intents. If a checkpoint is required due to the current buffer being full or any other activity such as a spin being requested, it schedules this by setting the appropriate return code.

Termination Exit

The Termination Exit posts the SYSOTCB subtask to shut down, and when this is complete, it detaches SYSOTCB.

SYSOTCB Subtask

Because of the restriction of the operating system that data sets must be opened and closed by the same task, a subtask is attached by the Initialization exit. It also performs the formatting and writing of any captured intent lists to the output data set, so that this process can overlap the writing of Cache data set by CHPTTASK.

The subtask initializes and then waits for one of the user exits to post it to perform a function. When the function is complete, the subtask posts an ECB to release the waiting user exit. The following functions are implemented:

OPEN	Allocate and open an output data set as specified.
CLOSE	Close the output data set. This also de-allocates the data set.
OFFLOAD	Format and write the formatted intent lists.
SHUTDOWN	If the output data set is open, close it and then terminate.

Sample Output

The sample output shows an intent list captured from an application that clears parameter set P1,P2,P3,P4 from table EXIT_SAMPLE and then inserts a number of rows into this table. This table is defined as shown in Sample Table Definition on page 179. The implemented user parameters are set as follows:

```
ZDDNAME=SAMP
ZTABLE=EXIT_SAMPLE
ZSTORE=9999
ZCLASS=*
```

Sample Job Log

```
09.23.07 JOB08272 ---- THURSDAY,  21 JUN 2007 ----
09.23.07 JOB08272  IRR010I  USERID USER2   IS ASSIGNED TO THIS JOB.
09.23.07 JOB08272  ICH70001I USER2   LAST ACCESS AT 09:21:32 ON THURSDAY, JUNE 21, 2007
09.23.07 JOB08272  $HASP373 DC50ADOB STARTED - INIT 17   - CLASS D - SYS STAR
09.23.08 JOB08272  -                                          --TIMINGS (MINS.)--            -----PAGING COUNTS----
09.23.08 JOB08272  -STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB  CLOCK   SERV  WORKLOAD  PAGE  SWAP   VIO SWAPS
09.23.08 JOB08272  -DELDUMP             00     49     25    .00    .00     .0    260  BATCH        0     0     0     0
09.23.08 JOB08272  -PARMS               00     52     26    .00    .00     .0    154  BATCH        0     0     0     0
09.23.09 JOB08272  -REDOPRT             00     72     44    .00    .00     .0    591  BATCH        0     0     0     0
09.23.09 JOB08272  USER EXIT SUBTASK ATTACHED
09.23.09 JOB08272  S6BKS312I USER EXITS HAVE BEEN ACTIVATED
09.23.09 JOB08272  S6BKS010I-DCA DATA OBJECT BROKER VERSION 5.0.123.
09.23.09 JOB08272  S6BKS011I-DCA TASK NAME DC50ADOB ASID 0158
09.23.09 JOB08272  S6BKS012I-DCA APPLID  S6DCDOBA NODENAME S6DCDOBA
09.23.09 JOB08272  S6BKS030I-DCA SYSTEM NON-SWAPPABLE
09.23.10 JOB08272  S6BDB067I-DCA RESIDENT PAGES REQUESTED = 080000, AVAILABLE = 080000
09.23.10 JOB08272  S6BKQ001I-DCA RTUQTASK INITIALIZATION COMPLETED SUCCESSFULLY
09.23.11 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.REDOLOG
09.23.11 JOB08272  S6BDB035I-DCA REDOLOG CI SIZE = 04096
09.23.11 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.CACHE1
09.23.11 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.CACHE2
09.23.11 JOB08272  S6BKA033I-DCA APPL INITIALIZATION OK (COPY 01)
09.23.11 JOB08272  S6BDB092I-DCA ACTIVATING SEGMENT=000 NAME=CRA.SEG0
09.23.11 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG0.PAGE1
09.23.11 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG0.PAGE2
09.23.12 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG0.PAGE3
09.23.12 JOB08272  S6BDB092I-DCA ACTIVATING SEGMENT=001 NAME=CRA.SEG1
09.23.12 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG1.PAGE1
09.23.12 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG1.PAGE2
09.23.13 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG1.PAGE3
09.23.13 JOB08272  S6BDB092I-DCA ACTIVATING SEGMENT=002 NAME=CRA.SEG2
09.23.13 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE1
09.23.13 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE2
09.23.13 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE3
09.23.14 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE4
09.23.14 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE5
09.23.15 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG2.PAGE6
09.23.15 JOB08272  S6BDB092I-DCA ACTIVATING SEGMENT=003 NAME=CRA.SEG3
09.23.15 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG3.PAGE1
09.23.15 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG3.PAGE2
09.23.16 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.SEG3.PAGE3
09.23.16 JOB08272  S6BDB092I-DCA ACTIVATING SEGMENT=099 NAME=CRA.S99
09.23.16 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.S99.PAGE1
09.23.16 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.S99.PAGE2
09.23.17 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.S99.PAGE3
09.23.17 JOB08272  S6BDB002I-DCA FILE INITIALIZATION OK
09.23.17 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.JRNL1
09.23.18 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.JRNL2
09.23.18 JOB08272  S6BKX068W-DCA CHTRANLIMIT SET BASED ON SPACE AVAILABLE IN THE REDOLOG DATA SE
09.23.18 JOB08272  S6BKX057W-DCA PAGE LIMIT REDUCED - EXCEEDS 15% OF RESIDENT PAGES
09.23.18 JOB08272  S6BKX060I-DCA CHECKPOINT MAXIMUM PAGES = 012000, TRANSACTIONS = 008999
09.23.18 JOB08272  VALUE OF ZSTORE RESET TO CHTRANLIMIT VALUE
09.23.18 JOB08272  S6BKS090I-DCA REDO SUBTASK DETACH (STAGE 2)
09.23.18 JOB08272  S6BKP001I-DCA OPER INITIALIZATION COMPLETED
09.23.18 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.RESOURCE
09.23.18 JOB08272  S6BDB090I-DCA OPEN S6B.DC.R50.CRA.REDOLOG.PENDING
09.23.22 JOB08272  S6BCM900I OPENPORT FOR COMMUNICATION IDENTIFIER "S6DCDOBA" HAS SUCCEEDED
09.23.22 JOB08272  S6BKC035L-DCA INITIATING PEER CONNECTION TO S6DCDOBB:S6DCDOBB
09.23.22 JOB08272  S6BRA008I TCP/IP RELAY started; Accepting connections on socket port 10021
09.23.22 JOB08272  S6BRA013E Connection to node "S6DCDOBB" refused
09.23.23 JOB08272  S6BKS050I-DCA SYSTEM READY  2007JUN21 09:23
```

Sample Recovered Intents

This is an output data set showing recovered intents captured by the Recovery user exit after the Data Object Broker was deliberately aborted.

Recovered Intent

```
00000000  00580001 000002E4 40000001 804ED200 000002E4 C5E7C9E3 6DE2C1D4 D7D3C540    *.˚.....U ...Ã+K....UEXIT_SAMPLE *
00000020  40404040 E3C4E240 00400000 00000000 00000000 00000000 00000000 00000000    *    TDS . ......................*
00000040  00000000 00000000 00000000 02D7F102 D7F202D7 F302D7F4 F000                 *.............P1.P2.P3.P40.      *
```

Recovered Intent

```
00000000  0001D900 0002EB00 020042C9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102 D7F202D7    *..R...ö...¡I.EXIT_SAMPLE.P1.P2.P*
00000020  F302D7F4 00290101 24E3C8C9 E240C9E2 40C140D9 D6E640D6 C640C4C1 E3C140C9    *3.P4.‰..„THIS IS A ROW OF DATA I*
00000040  D540E3C8 C9E240E3 C1C240D3 C5009DC9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102    *N THIS TAB LE.ÎI.EXIT_SAMPLE.P1.*
00000060  D7F202D7 F302D7F4 00840102 7FE3C8C9 E240C9E2 40E3C8C5 40E2C5C3 D6D5C440    *P2.P3.P4.d.."THIS IS THE SECOND *
00000080  D9D6E640 D6C640C4 C1E3C140 C9D540E3 C8C9E240 E3C1C2D3 C54B40C9 E340C9E2    *ROW OF DATA IN THIS TABLE. IT IS*
000000A0  40D7C1C4 C4C5C440 E6C9E3C8 40E3C8C5 40C3C8C1 D9C1C3E3 C5D940E7 40E3D640    * PADDED WITH THE CHARACTER X TO *
000000C0  E2C8D6E6 40C140E3 E6D640C2 E8E3C540 D3C5D5C7 E3C840C6 C9C5D3C4 4BE7E7E7    *SHOW A TWO BYTE LENGTH FIELD.XXX*
000000E0  E7E7E7E7 E7E7E7E7 E7E7E7E7                                                *XXXXXXXXXXXX                    *
```

Recovered Intent

```
00000000  0001D900 0002F000 020042C9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102 D7F202D7    *..R...0...¡I.EXIT_SAMPLE.P1.P2.P*
00000020  F302D7F4 00290103 24E3C8C9 E240C9E2 40C140D9 D6E640D6 C640C4C1 E3C140C9    *3.P4.‰..„THIS IS A ROW OF DATA I*
00000040  D540E3C8 C9E240E3 C1C240D3 C5009DC9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102    *N THIS TAB LE.ÎI.EXIT_SAMPLE.P1.*
00000060  D7F202D7 F302D7F4 00840104 7FE3C8C9 E240C9E2 40E3C8C5 40E2C5C3 D6D5C440    *P2.P3.P4.d.."THIS IS THE SECOND *
00000080  D9D6E640 D6C640C4 C1E3C140 C9D540E3 C8C9E240 E3C1C2D3 C54B40C9 E340C9E2    *ROW OF DATA IN THIS TABLE. IT IS*
000000A0  40D7C1C4 C4C5C440 E6C9E3C8 40E3C8C5 40C3C8C1 D9C1C3E3 C5D940E7 40E3D640    * PADDED WITH THE CHARACTER X TO *
000000C0  E2C8D6E6 40C140E3 E6D640C2 E8E3C540 D3C5D5C7 E3C840C6 C9C5D3C4 4BE7E7E7    *SHOW A TWO BYTE LENGTH FIELD.XXX*
000000E0  E7E7E7E7 E7E7E7E7 E7E7E7E7                                                *XXXXXXXXXXXX                    *
```

This is followed by similar entries for other inserts.

Sample Normal Intents

This is an output data set showing normal intents captured by the Redolog-1 user exit.

Normal Intent

```
00000000  00580001 000002E4 40000001 804ED200 000002E4 C5E7C9E3 6DE2C1D4 D7D3C540    *.˚.....U ...Ã+K....UEXIT_SAMPLE *
00000020  40404040 E3C4E240 00400000 00000000 00000000 00000000 00000000 00000000    *    TDS . ......................*
00000040  00000000 00000000 00000000 02D7F102 D7F202D7 F302D7F4 F000                 *.............P1.P2.P3.P40.      *
```

Normal Intent

```
00000000  0001D900 0002EB00 020042C9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102 D7F202D7    *..R...õ...¡I.EXIT_SAMPLE.P1.P2.P*
00000020  F302D7F4 00290101 24E3C8C9 E240C9E2 40C140D9 D6E640D6 C640C4C1 E3C140C9    *3.P4.%..„THIS IS A ROW OF DATA I*
00000040  D540E3C8 C9E240E3 C1C240D3 C5009DC9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102    *N THIS TAB LE.ÎI.EXIT_SAMPLE.P1.*
00000060  D7F202D7 F302D7F4 00840102 7FE3C8C9 E240C9E2 40E3C8C5 40E2C5C3 D6D5C440    *P2.P3.P4.d.."THIS IS THE SECOND *
00000080  D9D6E640 D6C640C4 C1E3C140 C9D540E3 C8C9E240 E3C1C2D3 C54B40C9 E340C9E2    *ROW OF DATA IN THIS TABLE. IT IS*
000000A0  40D7C1C4 C4C5C440 E6C9E3C8 40E3C8C5 40C3C8C1 D9C1C3E3 C5D940E7 40E3D640    * PADDED WITH THE CHARACTER X TO *
000000C0  E2C8D6E6 40C140E3 E6D640C2 E8E3C540 D3C5D5C7 E3C840C6 C9C5D3C4 4BE7E7E7    *SHOW A TWO BYTE LENGTH FIELD.XXX*
000000E0  E7E7E7E7 E7E7E7E7 E7E7E7E7                                                *XXXXXXXXXXXX                    *
```

Normal Intent

```
00000000  0001D900 0002F000 020042C9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102 D7F202D7    *..R...0...¡I.EXIT_SAMPLE.P1.P2.P*
00000020  F302D7F4 00290103 24E3C8C9 E240C9E2 40C140D9 D6E640D6 C640C4C1 E3C140C9    *3.P4.%..„THIS IS A ROW OF DATA I*
00000040  D540E3C8 C9E240E3 C1C240D3 C5009DC9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102    *N THIS TAB LE.ÎI.EXIT_SAMPLE.P1.*
00000060  D7F202D7 F302D7F4 00840104 7FE3C8C9 E240C9E2 40E3C8C5 40E2C5C3 D6D5C440    *P2.P3.P4.d.."THIS IS THE SECOND *
00000080  D9D6E640 D6C640C4 C1E3C140 C9D540E3 C8C9E240 E3C1C2D3 C54B40C9 E340C9E2    *ROW OF DATA IN THIS TABLE. IT IS*
000000A0  40D7C1C4 C4C5C440 E6C9E3C8 40E3C8C5 40C3C8C1 D9C1C3E3 C5D940E7 40E3D640    * PADDED WITH THE CHARACTER X TO *
000000C0  E2C8D6E6 40C140E3 E6D640C2 E8E3C540 D3C5D5C7 E3C840C6 C9C5D3C4 4BE7E7E7    *SHOW A TWO BYTE LENGTH FIELD.XXX*
000000E0  E7E7E7E7 E7E7E7E7 E7E7E7E7                                                *XXXXXXXXXXXX                    *
```

Normal Intent

```
00000000  0001D900 0002F500 020042C9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102 D7F202D7    *..R...5...¡I.EXIT_SAMPLE.P1.P2.P*
00000020  F302D7F4 00290105 24E3C8C9 E240C9E2 40C140D9 D6E640D6 C640C4C1 E3C140C9    *3.P4.%..„THIS IS A ROW OF DATA I*
00000040  D540E3C8 C9E240E3 C1C240D3 C5009DC9 0BC5E7C9 E36DE2C1 D4D7D3C5 02D7F102    *N THIS TAB LE.ÎI.EXIT_SAMPLE.P1.*
00000060  D7F202D7 F302D7F4 00840106 7FE3C8C9 E240C9E2 40E3C8C5 40E2C5C3 D6D5C440    *P2.P3.P4.d.."THIS IS THE SECOND *
00000080  D9D6E640 D6C640C4 C1E3C140 C9D540E3 C8C9E240 E3C1C2D3 C54B40C9 E340C9E2    *ROW OF DATA IN THIS TABLE. IT IS*
000000A0  40D7C1C4 C4C5C440 E6C9E3C8 40E3C8C5 40C3C8C1 D9C1C3E3 C5D940E7 40E3D640    * PADDED WITH THE CHARACTER X TO *
000000C0  E2C8D6E6 40C140E3 E6D640C2 E8E3C540 D3C5D5C7 E3C840C6 C9C5D3C4 4BE7E7E7    *SHOW A TWO BYTE LENGTH FIELD.XXX*
000000E0  E7E7E7E7 E7E7E7E7 E7E7E7E7                                                *XXXXXXXXXXXX                    *
```

This is followed by similar entries for other inserts.

Chapter 10 Using Multiple Data Object Brokers

This chapter describes how to setup and use multiple Data Object Brokers.

Topics

Overview

Multiple Data Object Brokers can be configured to address the same database. When implemented, one of these is designated the primary Data Object Broker, responsible for all access and updating of the database. Any other Data Object Broker that addresses this database is designated to be a secondary Data Object Broker and will be mostly idle until either the primary Data Object Broker fails, or an operator switches control to it. This mode of operating exploits the facilities of the IBM Coupling Facility and z/OS XCF communications.

In addition, a new TIBCO Object Service Broker address space called the Message Switch (MSW) helps to maintain access to the database by other TIBCO Object Service Broker components (such as Execution Environments and database servers) in the event of a switch of the primary Data Object Broker. Figure 1 shows a sample implementation.

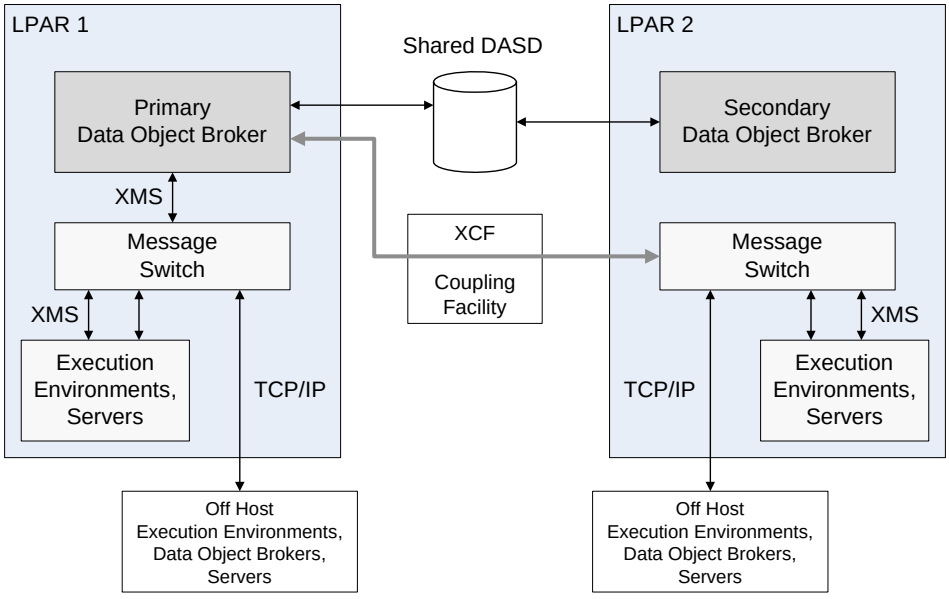


Figure 1 Multiple Data Object Brokers addressing a shared database

When running multiple Data Object Brokers, access to any Data Object Broker is only permitted via a MSW address space, and this access by MSW address spaces is only permitted using either the cross memory or XCF communication protocols. MSW address spaces do not support VTAM as a communication protocol.

In multiple Data Object Broker mode, it is the MSW address spaces that use what is currently the COMMID of a Data Object Broker in single mode. The Data Object Brokers automatically switch their communication port ID to be their XCF member name. As with a single Data Object Broker, the COMMID value for multiple Data Object Brokers must be specified for outgoing PEER-PEER connections.

Primary Data Object Broker Failure

When the primary Data Object Broker fails, all secondary Data Object Brokers and MSWs are notified by XCF. The MSWs terminate their sessions with other TIBCO Object Service Broker components and wait for the new primary Data Object Broker to complete initialization. Any connect requests from other TIBCO Object Service Broker components after this time will be held in the MSW address space until the primary Data Object Broker becomes available.

The first secondary Data Object Broker to obtain the system wide enqueue resource will become the new primary Data Object Broker, and will perform a standard TIBCO Object Service Broker restart, since there is no data available in the coupling facility. Once the new primary Data Object Broker becomes available, all MSW address spaces will process queued connection requests and processing will proceed as normal. User sessions will be able to reconnect to the primary Data Object Broker.

In addition, CICS Execution Environments, Native Execution Environments, and all z/OS gateways will automatically restart if the primary Data Object Broker fails. Only the active end-user sessions will not be reinstated. The above processing is intended to greatly reduce the outage time resulting from a Data Object Broker failure.

Configuring Data Object Brokers and Message Switches

A MSW address space should be started on each LPAR in a complex that requires access to a Data Object Broker. In this environment, Execution Environments and database servers on the same LPAR as a MSW will use cross memory communication to the MSW address space. The MSW address space will then use either cross memory communication, if the primary Data Object Broker is on the same LPAR, or the XCF communications protocol if it is not. However, this is not mandatory. Should Execution Environments or database gateways use either TCP/IP or XCF to communicate with its chosen MSW, and this MSW is not on the same LPAR as the primary Data Object Broker, system performance will be impacted.

Initialization

MSWs may be initialized before or after the Data Object Broker, but both the Data Object Broker and the MSW must be initialized before any other TIBCO Object Service Broker components attempt to connect. Since the MSW address space can wait for the primary Data Object Broker to become available, it is recommended that all MSW address spaces are initialized before the primary Data Object Broker. In addition, if a MSW is not available to the primary Data Object Broker when it initializes, outgoing peer server access will fail. Outgoing peer servers should have retry capability set in their resource details for this reason.

In the multiple Data Object Broker environment, Data Object Brokers and MSWs communicate via either XCF or XMS. It is suggested that the Data Object Brokers in this environment use individual HCS RELAY configuration files that enable XCF only. In addition, for access to MSWs via TCP/IP, it will be necessary to have individual HCS RELAY configuration files to enable them to use different IP addresses.

Configuring Data Object Brokers

This section describes the parameters used to enable multiple Data Object Brokers (the descriptions are provided here for convenience; also see *TIBCO Object Service Broker Parameters*).

Parameters

Parameter	Value	Description
XCFGROUP	A valid XCF group name	XCF group this Data Object Broker is to join. The presence of this parameter enables support of Multiple Data Object Brokers.
XCFMEMBER	A valid XCF member name, up 8 characters	Member name of this Data Object Broker instance in the XCF group, defaulted to the job name of this Data Object Broker instance.
XCFMODE	AUTOMATIC	This Data Object Broker becomes the primary if no primary Data Object Broker already exists in the XCF Group.
	SECONDARY	This Data Object Broker initializes as a secondary Data Object Broker.
XCFSTRUCTURE	Any valid structure name	Name of Coupling Facility structure to be used by this Data Object Broker group.

Example – Parameter Values

The following parameters initialize a system with three Data Object Brokers, where job name DC51ADB1 is the primary Data Object Broker. Unless shown, all other Data Object Broker parameters are identical as they would be for a normal configuration.

```
Parameter      Instance 1         Instance 2         Instance 3

JOBNAME        DC51ADB1           DC51ADB2           DC51ADB3
COMMID         S6DCDOBA           S6DCDOBA           S6DCDOBA
XCFGROUP       DCDOBA             DCDOBA             DCDOBA
XCFMEMBER      DCDOB1             DCDOB2             DCDOB3
XCFMODE        AUTOMATIC          SECONDARY          SECONDARY
XCFSTRUCTURE   S6BOSB_DCTDS_1     S6BOSB_DCTDS_1     S6BOSB_DCTDS_1
```

Sample Data Object Broker HCS Relay Parameter File

```
<relay xmlns='http://www.tibco.com/OSB/relayparms.xsd'>
  <xcfparms groupname='S6DCOSB'/>  <!--No TCP/IP support XCF only-->
  <directory>
    <node name="DCDOB1">         <!--Entries for multiple DOBs-->
      <xcf />
    </node>
    <node name="DCDOB2">
      <xcf />
    </node>
    <node name="DCDOB3">
      <xcf />
    </node>
    <node name="S6DCDOBA">       <!--Entry for the MSW-->
      <xcf />
    </node>
  </directory>
</relay>
```

Configuring Message Switches

This section describes the parameters used to configure a MSW (the descriptions are provided here for convenience; also see *TIBCO Object Service Broker Parameters*).

Parameters

Customize the MSRUN or MSRUNSTC JCL and the MSRUNPRM parameters. Most of the parameters are similar to their Data Object Broker counterparts.

Parameter	Value	Description
XCFGROUP	A valid XCF group name	XCF group this Data Object Broker is to join. The presence of this parameter enables support of multiple Data Object Brokers.
XCFMEMBER	A valid XCF member name, up to 8 characters	Member name of this Data Object Broker instance in the XCF group, defaulted to the job name of this Data Object Broker instance
XCFSTRUCTURE	A valid structure name	Name of Coupling Facility structure to be used by this Data Object Broker group.
XCFWAITTIME	0-43200	Number of seconds a MSW address space will wait for the primary Data Object Broker to become available. 0 = wait indefinitely.

In addition, the following Data Object Broker parameters are supported by Message Switches and have the same functionality as when applied to Data Object Broker:

- DESCACTION
- DESCERROR
- DESCINFORM
- DESCLOG
- DESCWARN
- INTERNALTRACE
- MAXUSERS
- ROUTACTION
- ROUTERROR
- ROUTINFORM
- ROUTLOG
- ROUTWARN
- WTOSUFFIX

Messages Switches also support the Execution Environment parameter TDS. When used to configure Messages Switches, TDS represents the COMMID of the Data Object Broker to which the MSW will connect.

Example – Parameter Values

Sample MSW parameters to connect to a Data Object Broker group:

```
MAXUSERS            1024
XCFGROUP            DCDOBA
XCFMEMBER           DCMSW1
XCFWAITTIME         60
TDS                 S6DCDOBA
```

Sample MSW HCS Relay Parameter File:

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
  <tcpipparms tcbnum="3" maxtcbsockets="1000" />
  <xcfparms groupname='S6DCOSB'/>
  <directory>
    <node name="S6DCDOBA">      <!--Entry for this MSW-->
      <xcf />
      <tcpip host="star.na.tibco.com" port="10021" />
    </node>
    <node name="DCDOB1">        <!--Entries for the multiple DOBs-->
      <xcf />
    </node>
    <node name="DCDOB2">
      <xcf />
    </node>
    <node name="DCDOB3">
      <xcf />
    </node>
    <node name="S6DCDOBB">       <!--Entry for a local Peer DOB-->
      <xcf />
    </node>
    <node name="DJCDOB">         <!--Entry for a remote Peer DOB-->
      <tcpip host="10.98.39.114" port="7249" />
    </node>
  </directory>
</relay>
```

Allocating the Coupling Facility Structure

To allocate the Coupling Facility structure, specify the name of Coupling Facility structure to be used by the Data Object Broker group, and its initial and maximum size, as follows:

```
STRUCTURE NAME(name)
    SIZE(maxsize)
    INITSIZE(initsize)
```

Parameter	Value	Description
STRUCTURE NAME	A valid structure name	Name of Coupling Facility structure. Must match the value of XCFSTRUCTURE used by the Data Object Broker group.
SIZE		Maximum size of the Coupling Facility.
INITSIZE		Initial size of the Coupling Facility.

Values for the initial and maximum size can be estimated using the S6BBRCFC utility. For details, see *TIBCO Object Service Broker Utilities*. Sufficient storage must be available in the specified Coupling Facility to allow key control blocks to be copied when an operator initiates a switch. For details on the switch process, see Switch Processing on page 202.

Managing Multiple Data Object Brokers

This section describes switch processing, and also lists the commands used to manage multiple Data Object Brokers and Message Switches.

Switch Processing

The following describes the processing that occurs when an operator initiates a switch from the primary Data Object Broker to a secondary Data Object Broker.

1. The operator enters a SWITCH command.
2. A checkpoint is initiated to reduce the restart time for the target Data Object Broker.
3. Once the checkpoint is complete, the primary Data Object Broker takes a systems-wide enqueue to stop any further changes to the Data Object Broker XCF Group. A target secondary Data Object Broker is chosen if the operator specified ANY on the switch command; otherwise, the specified secondary Data Object Broker will become the target Data Object Broker.
4. The primary Data Object Broker sets both Data Object Brokers into switch mode.
5. The secondary Data Object Broker performs a standard initialization. It will attempt to obtain the systems-wide enqueue currently held by the primary Data Object Broker, and will enter a wait until it becomes available.
6. All MSW address spaces drain message traffic to the current primary Data Object Broker. Messages currently active in the primary will be allowed to complete their processing, but new messages and session connections are queued in the MSW address space until the switch is complete.
7. Any transactions that are currently held by the Data Object Broker for locks are terminated with a LOCKFAIL condition.
8. Once all MSW address spaces connected to the primary Data Object Broker indicate that they are drained, the primary Data Object Broker initiates a shutdown.

9. As part of the shutdown, the following data is written to the Coupling Facility:
   - Status of all sessions connected to the primary Data Object Broker.
   - Status of resources in use by the primary Data Object Broker.
   - Status of all locks currently held on the primary Data Object Broker.
   - Status of the system that is usually retrieved from the REDOLOG data set upon initialization.
10. Once the primary Data Object Broker has written all the necessary data to the Coupling Facility and shuts down, it releases the enqueue obtained in step 3 above.
11. The selected secondary Data Object Broker comes out of the wait entered in step 5 and starts a standard TIBCO Object Service Broker initialization; however, instead of reading the REDOLOG data set for restart information, it reads all the data written to the Coupling Facility in step 9.
12. When the secondary Data Object Broker completes its initialization, it sets itself as the new primary Data Object Broker.
13. All MSW address spaces reconnect their active sessions to the new primary Data Object Broker and send any held messages and connections to the new primary Data Object Broker.

Operator Commands

The following commands are used to control multiple Data Object Brokers and Message Switches. For details, see Chapter 16, Operator Commands.

Data Object Brokers

- MEMBERLIST – displays the current status of all members of the Data Object Broker XCF group.
- SWITCH – controls switch processing.

Message Switches

These commands support Message Switch address spaces; their functions are the same as when applied to Data Object Brokers.

- CANCELUSER
- CANCELTERMID
- MEMBERLIST
- PARAMETER
- SHUTDOWN
- USERCOUNT
- USERLIST

Chapter 11 Using User Builtin Routines

This chapter describes how to customize your system with user builtin routines.

Topics

User Builtin Routines

User builtin routines are user-written assembler routines that are similar to external routines.

Refer to *TIBCO Object Service Broker for z/OS External Environments* for information about user builtin routines.

Installing a User Builtin Routine

Steps Required

Before installing a user builtin routine, refer to *TIBCO Object Service Broker for z/OS External Environments*.

To install a routine, complete the following steps:

1. Create a USRBLTIN member similar to the one in the ASM data set and include your own routine, preferably as an entry within the USRCSECT CSECT.
2. Add a $ROUTINE macro to the USRBLTIN CSECT.

   The USRBLTIN CSECT must contain only $ROUTINE macros. The parameters for the macro are:

NAME=*aaa*	*aaa* is the 1-16 character name that you code in a rule to invoke the routine.
MODULE=*bbb*	*bbb* is the 1-8-character CSECT or entry point name.
ARGS=*ccc*	*ccc* is the list of argument parameters in sets of four, that is, type, syntax, length, and decimals. For example, `ARGS=((,V,16,0),(,B,2,0))` defines a 16-byte variable string and 2-byte binary value. The TYPE specification of the arguments is left undefined. If the argument can be updated by the user builtin routine, the four parameters of the argument are preceded by a constant INOUT,,. An example is `ARGS=((INOUT,,V,16,0),(INOUT,,B,2,0))`. Note the two commas following the INOUT designation. A missing comma causes obscure execution-time errors.

ARGNAMES=*ddd*	*ddd* is the list of names of the arguments. For example, `(STRING,NUMBER)`.
RETURN=*eee*	This is specified only if the routine returns a value. If it does return a value, the format is the same as for ARGS, for example, `(,C,5,0)`, which is a five-byte string. The return value field, if specified, appears to the builtin routine as the first argument passed through the register 1 on entry to the routine. In our example, register 1 points to a list of addresses, as follows: register 1--> address of the return field, address of the string parameter field, and address of the NUMBER parameter field. If you do not specify RETURN=, only two parameters would appear in the list of addresses pointed to by register 1. The z/OS parameter-passing customer of marking the high order bit of the last address in the list is used. X'80' is OR-ed into the high-order byte of the last address.

3. Test the routine as an external routine, if possible.

   That way, you can remove most of the bugs without affecting your TIBCO Object Service Broker system or having to relink TIBCO Object Service Broker too often.

This is not possible if you use the $SAVE macro.

The routine does not have to be in USRCSECT, as described in step 1, but this avoids proliferation of CSECTs that are not removed from the load module if they are deleted from the usermod.

If you want to add a routine as a separate CSECT, call it USRxxxxx to avoid possible conflicts with current or future TIBCO Object Service Broker CSECTs.

Activating a Builtin Routine

To activate a routine, complete the following steps:

1. Select member USERMOD8 in the JCL.
2. Customize USERMOD8 by invoking the OSEMOD ISPF edit macro.
3. Submit the usermod. It should end with a RC=4.

Return code 4 is expected because the link edit warning messages IEW2454W, IEW2609W, and IEW2646W are normal and can be ignored.

Chapter 12 Starting and Shutting Down TIBCO Object Service Broker

This chapter describes how to start and to shut down TIBCO Object Service Broker.

Topics

Overview

Starting the TIBCO Object Service Broker System

The exact procedures you use to start TIBCO Object Service Broker depend upon the mix of TIBCO Object Service Broker components that you use and your z/OS system environment. In general, the steps are:

1. Start a Data Object Broker.
2. Start external database servers.
3. Start one or more Execution Environments.

Shutting Down the TIBCO Object Service Broker System

Shutting down the TIBCO Object Service Broker system involves the following steps:

1. Determine if users are logged in.
2. Cancel user IDs if necessary.
3. Shut down any servers that are running.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

Critical Messages

When TIBCO Object Service Broker encounters a serious problem, it issues to the operator a critical message, for example:

```
S6BTW005E  Highest level index updated for table "DJC_TAB1"
```

which indicates that the DJC_TAB1 table is almost full, and logs this message in the @CRITICALMSGS table. Critical messages are those that require action by the system administrator. Using the @CRITICALMSGS table, TIBCO Object Service Broker ensures that the messages are still available after they are gone from the operator's console.

It is the system administrator's responsibility to delete processed messages from the @CRITICALMSGS table. When the table is full, critical messages are issued to the operator console only.

Here is an example of the @CRITICALMSGS table showing the date and time when the messages it contains were issued:

```
BROWSING TABLE   :  @CRITICALMSGS
COMMAND ==>

                                                    SCROLL: P
      KEY        DATE     TIME   MESSAGE
_ ----------- -------- ------    -
_           1 20070527 152108     .
_           2 20070527 152944     .
_           3 20070527 154547     .
```

By browsing the second row, we see the issued message:

```
TABLE TYPE      : TDS
COMMAND ==>
------------------------------------------------------------------------------

KEY             :           2
DATE            : 20070527
TIME            : 152944
MESSAGE         : S6BTW005E Highest level index updated for table "DJC_TAB1"
                :
              :
```

Starting a Data Object Broker or External Database Server

Starting a Data Object Broker

Starting a Data Object Broker activates TIBCO Object Service Broker. Depending on your installation requirements, the Data Object Broker can operate as one of the following:

- Started task
- Batch job

Starting a Data Object Broker as a Started Task

To start a Data Object Broker as a started task, issue a **`Start`** command from the z/OS console, specifying the procedure name that was added to your system procedure library.

Sample JCL is the OSRUNSTC member in the JCL data set. This JCL starts TIBCO Object Service Broker as a started task. Its name and contents could be customized for your site.

Starting a Data Object Broker as a Batch Job

To start TIBCO Object Service Broker as a batch job, submit the JCL named OSRUN. OSRUN JCL is in the JCL data set. Messages similar to those shown for the started task startup appear in the batch job log. This JCL could have been relocated or renamed in your operating environment.

Restarting a Data Object Broker

TIBCO Object Service Broker automatically checks the journal cache to determine whether to perform a cold start or a warm start:

- If TIBCO Object Service Broker detects that the system was shut down normally, it performs a cold start. This starts the Data Object Broker with the segment availability as defined in the DBDLIB (specified by INIT=Y/N on the DB statement).
- If TIBCO Object Service Broker detects anything except a graceful shutdown, it automatically performs a warm start with recovery. This starts the Data Object Broker with the same segment availability that existed at the time of the abnormal termination.

- If TIBCO Object Service Broker cannot safely restart itself, it issues the appropriate messages to help you diagnose the problem.

- After a system failure, in order to synchronize all updates to the Pagestore that were in-flight at the time of the failure, you should restart TIBCO Object Service Broker and terminate it again using a normal shutdown. Failure to do this can result in a corrupted Pagestore if any offline processing is attempted on the Pagestore.
- The first time you start a Data Object Broker, the Data Object Broker assumes a previous failure; it issues message S6BKR006A and asks that you reply G or C to S6BKR098A. If you reply C, it cancels the start-up; if you reply G, it performs a cold start (no recovery processing).

Starting External Database Servers

External database servers are available to access database management systems outside TIBCO Object Service Broker. To learn which servers are available, refer to the Late Breaking News on our Support web site.

Each server type, depending on the external database to be accessed, has a unique startup procedure. Each server type must also be defined to the Data Object Broker using the Resource Manager facility in the Administration menu. Refer to 3. Resource Management on page 364 for additional information about this facility.

See Also

The *Service Gateway* manuals for information on starting the server for a specific DBMS.

The online *TIBCO Object Service Broker Messages With Identifiers* documentation for information about abnormal termination messages and the appropriate responses.

Starting an Execution Environment

After starting a Data Object Broker, you must start an Execution Environment under CICS or TSO, or as a Native Execution Environment. The different operating environments are discussed separately below. For details on Execution Environments, see *TIBCO Object Service Broker for z/OS External Environments*.

Parameters, usually set at installation time, determine operating characteristics of your Execution Environment. You can modify Execution Environment parameters through operator commands. For more information about operator commands refer to Chapter 16, Operator Commands, on page 259 and for a description of the parameters refer *TIBCO Object Service Broker Parameters*.

Starting a CICS Execution Environment

Benefits of the CICS Execution Environment

Starting a CICS Execution Environment enables you to run multiple TIBCO Object Service Broker sessions concurrently in the CICS environment. The benefit of using a CICS Execution Environment is that it reduces overhead resource consumption, and enables you to run significantly more user sessions on a given system.

Authorizing the CICS Execution Environment

The CICS Execution Environment runs as a z/OS subtask of CICS. To run this as an authorized subtask of CICS (enabling Cross Memory Services), install the TIBCO Object Service Broker SVC. Refer to Installing the TIBCO Object Service Broker SVC on page 52 for details.

Use the provided utility S6BSVCIN to install the SVC without requiring an IPL.

Starting the CICS Execution Environment Automatically

To bring up the CICS Execution Environment automatically when you start CICS, define the program **S6BCSINT** in the CICS startup PLT. If the startup PLT program detects that it cannot start the Execution Environment interface in an authorized state, a message prompts you to type one of the following values:

G	Go ahead and start the TIBCO Object Service Broker CICS interface in an unauthorized state.
C	Cancel or terminate the TIBCO Object Service Broker CICS interface initialization.

Establishing Sessions

When the initialization completes successfully, a message notifies you that the CICS Execution Environment subtask is attached. At this point users can, from a terminal in CICS, establish TIBCO Object Service Broker sessions with the Data Object Broker by entering the transaction code **HURN** or the transaction code assigned at installation time.

Restarting or Starting a CICS Execution Environment

If it is necessary to reestablish communications with the Data Object Broker after CICS is already initialized, complete the following steps:

Step 1: Define S6BCSINT

Make sure that the initialization program name **S6BCSINT** is defined to CICS. This enables communications to be reestablished with the Data Object Broker.

Step 2: Enter HINT Transaction Code

If you need to restart communications or you want to start a selected Execution Environment, enter the transaction code **HINT** from the z/OS system console or CICS session. Use this code as follows:

- From the terminal:

  HINT<Enter>

- From the z/OS system console:

  MODIFY cics_jobname,HINT<Enter>

Optional Parameters for HINT

You can specify the following optional parameters for **HINT**, in any order:

MDL=*value*	Specifies the pattern to allocate a VTAM ACB for the Execution Environment.
TDS=*value*	Specifies the Data Object Broker communications identifier to which you want to connect.
INSTLIB=*value*	Specifies the name of the installation library.
SYSLIB=*value*	Specifies the system library.

For More Information

For more information about customizing CICS transaction codes or defining programs to the CICS Initialization and Shutdown PLTs, refer to Installing the Service Gateway for CICS on page 87.

Starting a TSO Execution Environment

Sample EXECs for Starting a TSO Execution Environment

TIBCO Object Service Broker comes with sample REXX EXECs for starting a TSO Execution Environment as a user, a system administrator, and an operator. These EXECs are customized at installation time. For information on how these EXECs are customized, refer to Step 3: Customize TSO Execution Environment EXEC on page 56.

Example

For example, at your site you could be directed to use a customized copy of the USER EXEC that was renamed OSB. From a TSO session, type:

```
OSB<Enter>
```

The appearance of the TIBCO Object Service Broker workbench or initial application screen confirms that TIBCO Object Service Broker is ready for use.

Starting a Native Execution Environment

Benefits of the Native Execution Environment

By setting up a Native Execution Environment, you can avoid intermediate software layers (such as CICS or TSO) and improve response time. Multiple users can access the Data Object Broker concurrently through one Native Execution Environment. In an IMS TM environment, the IMS Message Processing Region that communicates with TIBCO Object Service Broker connects to a Native Execution Environment.

JCL for starting the Native Execution Environment

You can start the Native Execution Environment as either a started task or a batch job. Sample JCL is provided in the JCL data set as member NATIVEEE. For information on how this JCL can be customized, refer to Start a Native Execution Environment on page 84.

Logging In to TIBCO Object Service Broker

When the Native Execution Environment is running, users can log in to TIBCO Object Service Broker by referencing its VTAM application ID (applid) in their login statement as follows:

```
logon applid(applid) data(U='userid',P='password')
```

applid	The model applid (MDL=parameter value) can be defined using either of the following values: **1** – The MDL parameter value specified at TIBCO Object Service Broker installation time for the Native Execution Environments. **2** – MDL *value* specified by the JCL keyword PARM on the EXEC statement used to start the Native Execution Environment. Parameters specified in the EXEC statement override the defaults coded at TIBCO Object Service Broker installation time. This can be any valid applid that is not already used for a pool of TIBCO Object Service Broker applids or for Data Object Brokers.
userid	TIBCO Object Service Broker user ID

password	TIBCO Object Service Broker password *Optional*: If P=*password* is not specified, the user is prompted for a password.

Including Other Execution Environment Parameters

Other Execution Environment session parameters can be included in the login statement to a maximum of 64 characters.

See Also *TIBCO Object Service Broker for z/OS External Environments* for more information about Execution Environments.

TIBCO Object Service Broker Parameters for more information about Execution Environment and session parameters.

Shutting Down the TIBCO Object Service Broker System

Overview

To ensure an orderly shutdown, complete the following tasks:

1. Determine Who Is Logged In to TIBCO Object Service Broker, page 219
2. If Necessary, Cancel Users, page 220
3. Shut Down the Servers, page 222
4. Shut Down the Execution Environments, page 222
5. Shut Down the Data Object Broker, page 224

Each of these tasks is described in detail below.

Task A Determine Who Is Logged In to TIBCO Object Service Broker

There are several ways to determine which users are currently logged in to TIBCO Object Service Broker:

From within TIBCO Object Service Broker

From the administrator workbench, position your cursor beside the `ST Statistics` option and press Enter. This selection displays the TIBCO Object Service Broker Statistics main menu. You can then select the `Users Logged On` option and press Enter.

From the TIBCO Object Service Broker Administration Menu

Execute the S6BTLADM utility and use option I. Refer to I. User Activity on page 325 for more information.

From the z/OS Operator Console

If you have access to a z/OS operator console, you can issue the following command:

```
MODIFY dob_jobname,USERLIST
```

In this command, *dob_jobname* is the name of the batch job or started task used to start the Data Object Broker. For more information about the forms of this command, refer to Chapter 16, Operator Commands, on page 259.

From Within CICS

Use the **HINQ** transaction to list users currently active to the CICS Execution Environment. The **HINQ** transaction states whether the Execution Environment subtask is attached or detached, and lists the number of users and their user IDs.

This transaction code is issued in the following form:

```
HINQ USERC
```

This is the same as above without listing the user IDs

You can also issue the **HINQ** command as part of a **Modify** command from the z/OS operator console. For example:

```
Modify cics_jobname,HINQ userid
```

Refer to Commands to Control the CICS Execution Environment on page 279 for more information about the **HINQ** command. For more information about issuing TIBCO Object Service Broker operator commands, refer to Chapter 16, Operator Commands, on page 259.

Do *not* use the CICS message utility **CMSG** to broadcast a message telling all users to log out. **CMSG** overwrites the screens of CICS users who are in pseudo-conversational mode.

Task B If Necessary, Cancel Users

If you need to cancel a specific user ID, do one of the following:

From the Data Object Broker

From the z/OS operator console, issue the command:

```
Modify dob_jobname,Canceluser=userid
```

where

dob_jobname	The name of the batch job or started task used to start the Data Object Broker.
userid	The user ID you want to cancel.

With the TIBCO Object Service Broker Administration Menu

Execute the S6BTLADM utility and use option I. Refer to I. User Activity on page 325 for more information.

From the Execution Environment

From the z/OS operator console, issue the command:

```
Modify ee_jobname,HCUS userid
```

where

ee_jobname	The name of the batch job or started task used to start the Execution Environment.
userid	The user ID you want to cancel.

If multiple connections have the same user ID, a list of all connections for the specified user ID is presented on the console. Select the connection you require and reenter the command with the appropriate reference number extension.

From the CICS Execution Environment

Do one of the following:

- From any terminal that is in session with CICS, enter the **HCUS** (**Canceluser**) command as follows:

  ```
  HCUS userid
  ```

- From the z/OS operator console, issue the **HCUS** (**Canceluser**) command as follows:

  ```
  Modify cics_jobname,HCUS userid
  ```

cics_jobname	The name of the CICS Execution Environment.
userid	The user ID you want to cancel.

Task C Shut Down the Servers

All users and servers are automatically logged out if they are still active when the Data Object Broker begins to shut down. The Data Object Broker also automatically issues final SMF records at this time.

You should use the **STOPSERVER** command to shut down any batch servers and external database servers before you initiate the Data Object Broker shutdown.

- Shutting down the batch servers is described in Chapter 14, Managing Batch Processing, on page 241.
- Shutting down the external database servers is described in the appropriate Service Gateway manual.

Shutting Down Servers Using the Administration Menu (S6BTLADM)

Execute the S6BTLADM utility and use option 3 to shut servers down using the Resource Management of the Administration menu. Refer to 3. Resource Management on page 364 for more information.

Shutting Down Servers Using the z/OS Operator Console

If you have access to a z/OS operator console, you can issue the following command:

```
MODIFY dob_jobname,STOPSERVER=connection_id
```

In this command, *dob_jobname* is the name of the Data Object Broker batch job or started task addressed by this command and the *connection_id* is the unique identifier for the connection. For valid values for this command, refer to Chapter 16, Operator Commands, on page 259.

Task D Shut Down the Execution Environments

This section describes how to shut down the various types of Execution Environments in the TIBCO Object Service Broker environment.

TSO

A TSO Execution Environment shuts down when its user logs out of TIBCO Object Service Broker.

CICS

Do one of the following:

- To shut down a CICS Execution Environment without shutting CICS down, enter the transaction code **HTRM** from any CICS session or from the z/OS operator console.

  Issue the following command from the console to delay the shutdown until active users are done:

  MODIFY cics_jobname,HTRM SHUTD

  Issue the following command from the console for an immediate shutdown:

  MODIFY cics_jobname,HTRM SHUTI

- To shut down the CICS Execution Environment automatically when you shut down CICS, define the program S6BCSTRM to the CICS shutdown PLT.

Native Execution Environment

You can shut down a Native Execution Environment by doing one of the following:

- Execute the S6BTLADM utility and use option I. Refer to I. User Activity on page 325 for more information.
- From the z/OS operator console, ensure that all users are logged out and then issue one of the following commands:

  p *ee_jobname*

  or

  Modify *ee_jobname*,**Shutdown**

  where

P	z/OS operator command (**Stop**).
Modify	z/OS operator command (can be abbreviated to **F**).
ee_jobname	The name of the batch job or started task used to start the Native Execution Environment.

Refer to Commands to Control the Native Execution Environment on page 275.

Task E Shut Down the Data Object Broker

You can shut down the Data Object Broker by doing one of the following:

- Execute the S6BTLADM utility and use option I. Refer to I. User Activity on page 325 for more information.
- Use option 2 (OPERATOR FUNCTIONS) of the S6BTLADM utility to access the **`Shutdown`** command. Refer to 2. Operator Functions on page 363 for more information. The S6BTLADM utility is described in Chapter 18, Monitoring and Controlling the TIBCO Object Service Broker Environment, on page 287.
- Issue the **`Shutdown`** command from the z/OS operator console:

  `Modify` *dob_jobname*`,`**`Shutdown`**

  In this example, *dob_jobname* is the name of the batch job or started task used to start the Data Object Broker. Active connections to the Data Object Broker are terminated when the **`Shutdown`** command is processed.

Unlike other TIBCO Object Service Broker operator commands, the **`Shutdown`** command *cannot* be abbreviated.

Chapter 13 Managing TIBCO Object Service Broker Networked Resources

This chapter describes how to manage TIBCO Object Service Broker networked resources.

Topics

Overview

What is a TIBCO Object Service Broker Network?

A TIBCO Object Service Broker network is a system of interconnected Data Object Brokers. A connection is a data transfer path between two specific components. Each Data Object Broker in a TIBCO Object Service Broker network carries its own view of the portion of the network in which it participates. From a specific Data Object Broker's view of the network, all other components are remote nodes (or *peer* Data Object Brokers), with the central Data Object Broker being the local node.

Allowed Connections to the Data Object Broker

You can connect the following components to the local Data Object Broker:

- Another Data Object Broker
- A TIBCO Object Service Broker Peer Server
- A TIBCO Service Gateway for Files
- A TIBCO Object Service Broker external database gateway

Making Remote Components Accessible

The DYNAMICRESOURCE Data Object Broker Parameter setting determines which external resources must be defined in the Resource Management Repository File.

- If DYNAMICRESOURCE is set to N, then all external resources must be defined in the Resource Management Repository File, as shown in the illustrations in Description of a Sample Network on page 231.
- If DYNAMICRESOURCE is set to Y, then it is only necessary to define the HRN and HIN entries describing the PEER to PEER connections. See S6BDYNAMIC Schedule Entry on page 229.

Resource Management Repository File

Overview

The local Data Object Broker's view of the network is contained in its resource management repository file, which is a VSAM Keyed Sequenced Data Set (KSDS) named Q1.Q2.RESOURCE. The resource management repository file is maintained using the Resource Management option of the Administration menu.

In a network that includes more than one Data Object Broker, it is not possible for a single file to describe the total configuration. The total TIBCO Object Service Broker network configuration can, in this case, be determined by looking at the resource files for all the individual Data Object Brokers.

As with any KSDS data set, the efficiency of the file decreases as maintenance is performed. Use IDCAMS utilities to reorganize the file.

Resource Management Option

To maintain the repository file, use the Resource Management option (option 3) of the Administration menu. The following illustration shows the initial RESOURCE MANAGEMENT screen for a Peer Server connection that uses default values. Refer to Chapter 18, Monitoring and Controlling the TIBCO Object Service Broker Environment, on page 287 for information about the Administration menu and to 3. Resource Management on page 364 for detailed information about using the Resource Management option.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR API DEFAULT0     2006NOV29 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE     Y LAST USER REUSE     Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES       0
 NODE  AMDSDCTHURON     INDOUBTS          N MONITOR/SMF         N

                        CONNECTIONS      IN-USE         TRX          MESSAGE
                       CUR  MAX  LMT   CUR  MAX       COUNT          COUNT
            ONLINE       0    0    0     0    0          0              0
            COMMON      10   10   10     0    0          0              0

 SCHEDULE NAME COMMON

           APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN     DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                 00:00    N          10




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
 NO MORE DATA TO BE DISPLAYED
```

S6BDYNAMIC Schedule Entry

If the schedule entry S6BDYNAMIC is not present in the Resource Management Repository File, a new entry is created there when the Data Object Broker is initialized.

The initial values for this schedule are show below:

```
S6BADM37  S6DCDOBA     RESOURCE SCHEDULE S6BDYNAMIC
2011OCT20 07:20:55

 ACT       APPLICABLE DAYS         EXCEPTION START  ONLINE
CONNECTIONS
     MON TUE WED THR FRI SAT SUN     DATE      TIME    ONLY        MAX
      Y   Y   Y   Y   Y   Y   Y              00:00    N           32









                        TARGET RESOURCE ___ _________
ENTER-DETAIL PF2-SCH IX PF4-COPY PF9-NEW PF10-UPDATE
DISPLAY FROM TOP OF DATA
```

These values may be modified using the Administration Menu, but the entry may not be deleted and is reserved for the exclusive use of dynamically created resources.

If the schedule S6BDYNAMIC satisfies the needs of any external connections to the Data Object Broker, only the PEER HRN and HIN entries need to be added to the Resource Management Repository File.

If the relevant permanent or dynamic resource entry exists, an external resource uses the existing entry when connecting to the Data Object Broker. If no entry exists, an entry matching the external resource's requirements is created dynamically. This dynamically created resource entry will be deleted as follows:

- When the use count for the entry goes to 0.
- When the Data Object Broker is initialized, unless there are external resources that have outstanding IN*DOUBT transactions associated with them.

The entries added dynamically may not be modified or deleted using the Administration Menu.

Repository File Maintenance

Online Backups

To save a Data Object Broker's resource repository file, use the S6BTLBRM (Resource Management Online Backup) utility. The utility writes out a copy of all resource detail and resource schedule entries to a sequential file, which can serve as a backup of the configuration.

Offline Backups

To back up a Data Object Broker's resource repository file while the Data Object Broker is offline, use standard IDCAMS utilities.

See Also *TIBCO Object Service Broker for z/OS Utilities* for information about the S6BTLBRM utility.

Sample TIBCO Object Service Broker Network Configurations

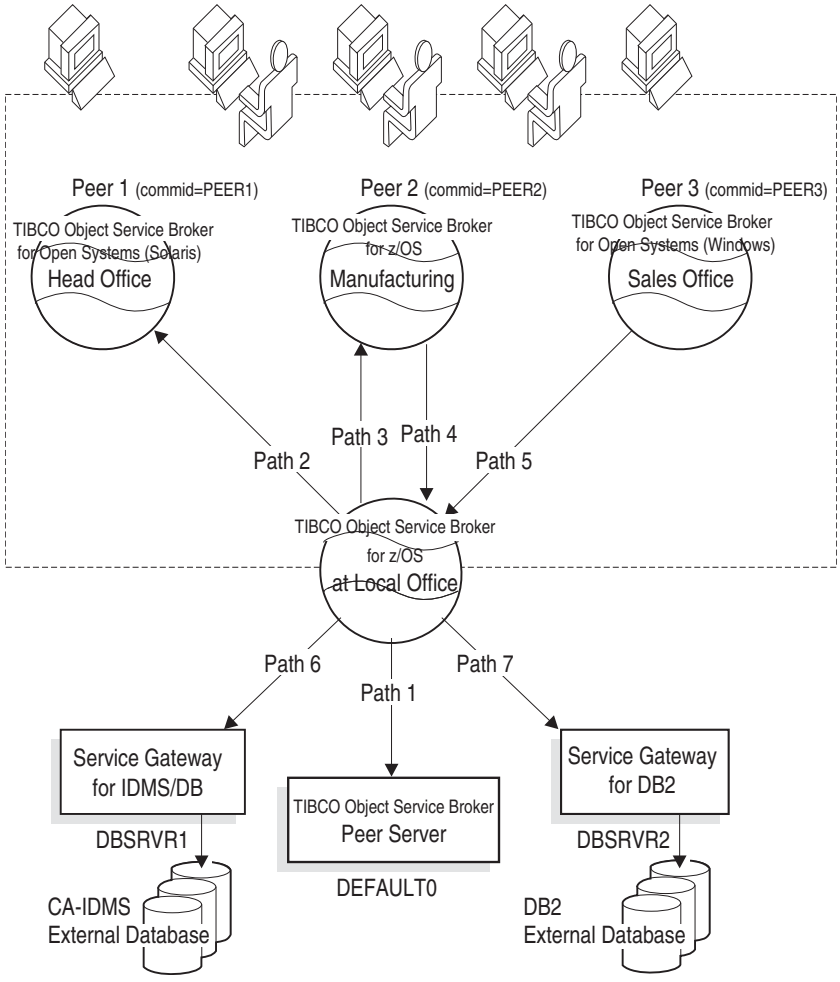
Description of a Sample Network

The following sample TIBCO Object Service Broker network subset revolves around the Local Office node that runs on z/OS. Although the two external databases and the TIBCO Object Service Broker peer rules server are called *remote* nodes from a networking perspective, they actually reside in the same z/OS operating system image as the local node.

The peer TIBCO Object Service Broker nodes are all physically remote. One also operates on z/OS, on a physically separate system from the local node. The other two operate on the Windows and Solaris operating systems.

Illustration of a TIBCO Object Service Broker Network

The following illustrates the sample TIBCO Object Service Broker network:



Resource Detail Entries

The resource detail entries for the sample network are shown in the following table. The schedule name for these resource entries is `WEEKDAYS05`.

Resource Type	Resource Name	Intermediate Rollback	Early Release	Last User	Commit Level	Node Name
API	DEFAULT0[a.]	Y	Y	Y	2	
HRN	PEER1 [b.]	Y	Y	N	2	HEADOFFICE
HRN	PEER2[b.]	Y	Y	N	2	MANUFACTURING
HIN	PEER2[b.]	Y	Y	N	2	MANUFACTURING
HIN	PEER3[b.]	Y	Y	N	2	SALESOFFICE
IDM	DBSRVR1	N	Y	N	1	
DB2	DBSRVR2	Y	Y	Y	1	

a. Resource type of API can represent either a Peer Server or a Service Gateway for Files. You must define different resource names for a Service Gateway for Files and a Peer Server. Do not use the name DEFAULT0 for a Service Gateway for Files.

b. For resource types of HRN and HIN, the resource name must be the COMMID of the remote Data Object Broker.

Resource Schedule Entries

The resource schedule entries for the sample network are as follows:

Schedule Name	Applicable Days (MTWTFSS)	Exception Date	Start Time	Online Only	Maximum Connection Count
WEEKDAY05	Y Y Y Y Y N N	2000JAN31	0600	N	5
	Y Y Y Y Y N N		0800	N	0

Description of Resources

Resource 1: API DEFAULT0

- Connects to a TIBCO Object Service Broker Peer Server.
- The Default schedule for type API is in effect for this resource. Refer to Resource Management Option on page 228 for an example of this resource. The Default schedule specifies a maximum of 16 resources, therefore 16 resources in this group are enabled starting at 5:00 a.m. seven days weekly, without overriding schedule entries.

Because API can represent either a Peer Server or a Service Gateway for Files you must ensure that the name for a Service Gateway for Files differs from the name of a Peer Server.

```
S6BADM33   S6H1DOBB      RESOURCE DETAIL FOR API DEFAULT0       2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE     Y LAST USER REUSE     Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES       0
 NODE                   INDOUBTS          N MONITOR/SMF         N

                       CONNECTIONS      IN-USE         TRX          MESSAGE
                      CUR   MAX   LMT   CUR   MAX      COUNT          COUNT
            ONLINE      0     0     0     0     0          0              0
            COMMON      0     0    16     0     0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
     MON TUE WED THR FRI SAT SUN   DATE        TIME   ONLY       MAX
      Y   Y   Y   Y   Y   Y   Y                05:00    N          16




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
 NO MORE DATA TO BE DISPLAYED
```

Resource 2: HRN PEER1

Connects to a remote TIBCO Object Service Broker node, the Head Office, for outgoing distributed data accesses.

The term outgoing does not refer to data flow direction. Data flows in both directions on this resource. It means that applications in the local node can instigate outgoing distributed data accesses to the remote node and it initiates the connection request.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR HRN PEER1         2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE    Y LAST USER REUSE    Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME            USER ID PREFIX   PRD FAILURES      0
 NODE  HEADOFFICE       INDOUBTS         N MONITOR/SMF        Y

                       CONNECTIONS      IN-USE         TRX          MESSAGE
                      CUR  MAX  LMT    CUR  MAX       COUNT          COUNT
           ONLINE       0    0    0     0    0          0              0
           COMMON       0    0   25     0    0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00   N          25




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Resource 3: HRN PEER2

- Connects to another remote TIBCO Object Service Broker node (at the Manufacturing location) for outgoing remote data accesses.
- No scheduling entries are explicitly defined for this resource group, therefore by default it uses the DEFAULT0 resource type schedule for an HRN resource type.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR HRN PEER2         2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE     Y LAST USER REUSE     Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES       0
 NODE  MANUFACTURING    INDOUBTS          N MONITOR/SMF         N

                        CONNECTIONS      IN-USE         TRX          MESSAGE
                       CUR  MAX  LMT    CUR  MAX       COUNT          COUNT
            ONLINE       0    0    0     0    0           0              0
            COMMON      10   10   10     0    0           0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00    N          10




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Resource 4: HIN PEER2

- This is for incoming remote data accesses from the same remote TIBCO Object Service Broker node at the Manufacturing location as Resource 3.
- No scheduling entries are explicitly defined for this resource group, therefore by default it uses the DEFAULT0 resource type schedule for an HIN resource type.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR HIN PEER2         2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE    Y LAST USER REUSE    Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME            USER ID PREFIX       FAILURES      0
 NODE  MANUFACTURING    INDOUBTS          N MONITOR/SMF        Y

                        CONNECTIONS      IN-USE         TRX          MESSAGE
                       CUR  MAX  LMT    CUR  MAX       COUNT         COUNT
            ONLINE       0    0    0     0    0          0             0
            COMMON       0    0   25     0    0          0             0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00   N          25




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Resource 5: HIN PEER3

- This connection accommodates incoming data access requests from the remote TIBCO Object Service Broker node, the Sales Office.

  Since this resource is specifically for incoming requests, the local node is unable to send data access requests to the remote node. Any data flow in this resource must be initiated from the remote node.

- No scheduling entries are explicitly defined for this resource group, therefore by default it uses the DEFAULT0 resource type schedule for an HIN resource type.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR HIN PEER3     2007MAY11    09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE     Y LAST USER REUSE     Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES       0
 NODE  SALESOFFICE       INDOUBTS          N MONITOR/SMF         Y

                        CONNECTIONS      IN-USE         TRX          MESSAGE
                       CUR  MAX  LMT    CUR  MAX       COUNT          COUNT
            ONLINE       0    0    0     0    0          0              0
            COMMON       0    0   10     0    0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
     MON TUE WED THR FRI SAT SUN   DATE        TIME   ONLY       MAX
      Y   Y   Y   Y   Y   Y   Y                00:00   N          10




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
 NO MORE DATA TO BE DISPLAYED
```

Resource 6: IDM DBSRVR1

- The Service Gateway for IDMS/DB is connected to the local Data Object Broker. If CAIDMS data is required by the manufacturing or sales node, a distributed request is routed to the Service Gateway for IDMS/DB via the local Data Object Broker.
- This resource group has its own unique schedule entry with type = IDM and identifier = DBSRVR1.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR IDM DBSRVR1     2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE    Y LAST USER REUSE    N COMMIT LEVEL  0
 RETRY INTERVAL       0 TP NAME            USER ID PREFIX       FAILURES      0
 NODE                   INDOUBTS         N MONITOR/SMF        Y

                       CONNECTIONS      IN-USE         TRX          MESSAGE
                      CUR  MAX  LMT   CUR  MAX       COUNT          COUNT
            ONLINE      0    0    0     0    0          0              0
            COMMON      0    0   10     0    0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00    N          10




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Resource 7: DB2 DBSRVR2

- Data accesses directed to the DB2 database from applications hosted by the local node, the Manufacturing Office, or the Sales Office are relegated to this resource, which connects to the DB2 database.
- This resource group has its own unique schedule entry with type = DB2 and identifier = DBSRVR2.

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR DB2 DBSRVR2     2007MAY11 09:16:22

 INTERMEDIATE ROLLBK  Y EARLY RELEASE    N LAST USER REUSE    N COMMIT LEVEL  1
 RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES      0
 NODE  INDOUBTS         N MONITOR/SMF         N

                       CONNECTIONS      IN-USE         TRX          MESSAGE
                      CUR  MAX  LMT    CUR  MAX       COUNT          COUNT
           ONLINE       0    0    0     0    0          0              0
           COMMON      10   10   10     0    0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00    N          10




 ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Chapter 14 Managing Batch Processing

This chapter describes how to manage batch processing.

Topics

Overview

What is a Batch Server?

A batch server is a job that manages a job queue. A batch server services a single queue; if you require several batch queues at your site, you need several batch servers to service them. Jobs are placed in job queues using the BATCH tool.

Running a Batch Server

A batch server can be run as one of the following:

- Started task
- Batch job

You should set up the batch server to run as a batch job. If you decide to run the batch server as a started task, you should ensure that jobs submitted by the batch server can obtain the necessary security authorization to run. The batch server dynamically allocates an internal reader to submit jobs. The requests in the queue are serviced in a synchronous manner and if for any reason the batch server cannot obtain an internal reader, the server delays servicing the requests on its queue until an internal reader is available.

Prerequisites

Before you can use the batch server for the first time, you must customize the following two table instances of the @BATCH_JCL table:

- @BATCH_JCL(@DEFAULT,HURON)
- @BATCH_JCL(@DEFAULT,JOBCARD)

We recommend that you customize this JCL by using the TIBCO Object Service Broker text editor (TED). Full customization instructions are included as a post-installation step in Customization of the TIBCO Object Service Broker @SCHEDULEMODEL Table on page 70.

Only specially authorized users belonging to the security group ADMIN can customize the @BATCH_JCL table instances and start batch servers.

See Also *TIBCO Object Service Broker Shareable Tools* for more information about the BATCH and TED tools.

Starting and Shutting Down the Batch Server

Starting the Batch Server

Since the batch server functions as an Execution Environment, the Data Object Broker must be online when the server is started. JCL for starting the batch server is defined at installation time, as described in Members OSRUNSTC and OSRUN in Data Object Broker JCL contain a SYSMDUMP DD statement. That data set is part of a GDG defined in STEP6 of member S6A3ALOC in the OSB.JOBS data set with a limit of 5, which is defined by the OSEMOD variable $DMPGDG$. on page 69. The BATSRVL1 member of the JCL data set contains JCL to run a batch server as a batch job.

Shutting Down the Batch Server

To shut down the batch server manually, issue one of the following commands from the z/OS operator console:

```
Stop jobname
```

or

```
Modify jobname,Stop
```

where

Modify	The z/OS operator command.
jobname	The name of the batch server job.
Stop	The z/OS operator command.

Invoking the Batch Submission Facility Menu

Execute BATCH

To review an existing batch queue or define a new batch queue, log in to the workbench and execute the BATCH tool by using the EX Execute Rule option and typing BATCH, or by moving the cursor to the primary command field and typing:

```
COMMAND ==> EX BATCH
```

When you press Enter, the Batch Submission Facility main menu appears as described in the following section.

Batch Submission Facility Main Menu

Illustration of the Batch Submission Facility Menu

```
                    Batch Submission Facility


              Submit Batch Request _

              Build Batch Commands _

                 View Batch Status _

                  Queue Definition _









PFKEYS: 2=LOGS 3=EXIT 12=EXIT
```

Menu Options

The first three options on this menu are user options that you can use to submit batch jobs, customize user JCL, and view the status of batch jobs.

You can use the fourth option on the main menu, Queue Definition, to review the definition of existing queues or define new queues. This option is described in more detail in the following sections.

See Also

The BATCH tool in *TIBCO Object Service Broker Shareable Tools* for more information on the options on the Batch Submission Facility menu.

Reviewing Existing Queues

Step 1: List Queue Definitions

To list queue definitions, position your cursor beside the Queue Definition option on the main menu and press Enter. A screen similar to the one below displays a list of queues that are currently defined in the system.

Screen Showing List of Queues Available

This example shows four existing queues: AFTERNOON, IMMEDIATE, OVERNIGHT, and SUNDAY.

```
                        Batch Submission Facility
                            Queue Definitions

 < Define New Queue >  ___________________


              AFTERNOON  _
              IMMEDIATE  _
              OVERNIGHT  _
                 SUNDAY  _









PFKEYS: 1=HELP 3=EXIT 12=EXIT
```

Step 2: View Queue Definitions

To view the definition of a queue, position your cursor on the field beside the name of that queue and press Enter. The definition of the queue appears as shown in the following illustration:

Screen Showing Queue Definition

```
                        Batch Submission Facility


                   Queue Definition for: SUNDAY


Wait Duration ==> 15              (# of seconds to wait when queue empty)
Wait Limit    ==> 12              (# of waits before queue shut down)


Default Output Settings:
 Print Destination ==> PRINTER2             Print Form      ==>
 Print Class       ==> Y                    External Writer ==>
 Number of Copies  ==> 1                    FCB             ==>
                                            UCS             ==>


Description:



PFKEYS: 1=HELP 3=SAVE 12=CANCEL 22=DELETE
```

Fields on the Queue Definition Screen

The fields on this screen are as follows:

Wait Duration	The time in seconds that the batch server waits before rescanning the queue when the queue becomes empty. You can enter a number from 0 to 32400 (9 hours).
Wait Limit	The number of times the batch server goes into a wait state before shutting itself down. You can enter a number from 0 to 9999.

Default Output Settings

The default print option values are used when the user submitting a batch request does not supply those values on the job submission screen.

Print Destination	The name of the printer where you want to send the batch job output. If you want to include the node name of the printer, specify the print destination in the form *nodename.printerid*.
Print Class	The class of the output
Number of Copies	The number of output copies desired
Print Form	Specifies that the output data should be printed on a special output form
External Writer	If you want to direct system output to an unsupported device, you can specify an external routine to direct the output.
FCB	Forms Control Block—specifies the type of output format
UCS	Universal Character Set—describes the character set that should be used for printing the output data set
Description	A brief description explaining the purpose of this queue

Defining New Queues

Step 1: Enter the Name of the New Queue

To enter a new queue name, complete the following steps:

1. Position your cursor beside the Queue Definition option on the main Batch Submission Facility menu and press Enter.

   The screen displays a list of queues that are currently defined in the system.

2. Type in a unique queue name in the field Define New Queue.

   The screen shown below illustrates how an ADMIN queue can be defined:

```
                        Batch Submission Facility
                            Queue Definitions

 < Define New Queue >  ADMIN_____________


              AFTERNOON  _
              IMMEDIATE  _
              OVERNIGHT  _
                 SUNDAY  _









PFKEYS: 1=HELP 3=EXIT 12=EXIT
```

3. Press Enter.

   The Queue Definition screen appears.

Step 2: Enter Values for the Definition

Enter values for the new queue. The ADMIN queue could be defined on the Queue Definition screen as shown below:

```
                        Batch Submission Facility


                    Queue Definition for: ADMIN


Wait Duration ==> 60              (# of seconds to wait when queue empty)
Wait Limit    ==> 120             (# of waits before queue shut down)


Default Output Settings:
 Print Destination ==> PRINTER2             Print Form      ==>
 Print Class       ==> Y                    External Writer ==>
 Number of Copies  ==> 1                    FCB             ==>
                                            UCS             ==>


Description:



PFKEYS: 1=HELP 3=SAVE 12=CANCEL 22=DELETE
```

In this example, the Wait Duration is set to 60 seconds. This means that when the queue becomes empty, the batch server waits 60 seconds before rescanning the queue. The Wait Limit is set to 120. This means that the batch server can go into a wait state up to 120 times before the batch server shuts itself down. If you start this queue at 8 a.m., these settings ensure that the queue would submit jobs and check for more jobs for a minimum of 2 hours, until 10 a.m.

The fields on this screen are described in Fields on the Queue Definition Screen on page 247.

Step 3: Save the New Queue Definition

To save this new queue definition, press PF3.

Before you can submit a job to the new queue, you must define a batch server to service the new queue. For information about defining the batch server, refer to Defining Additional Batch Servers on page 251.

Defining Additional Batch Servers

Step 1: Copy the JCL for an Existing Server

Refer to the JCL in Starting the Batch Server on page 243.

Step 2: Modify Input Parameters in the Startup Parameter Member

The input parameters referenced in the //BPARM statement come from a member specified by *$BATSRI$*. This variable is replaced with XBATCHL1. Modify the start-up parameter member (for example, create XBATCHL2, XBATCHL3, and so on). The input parameters in the member should appear as shown below:

```
USERID=OSB_USERID
TDS=$HURSRV$
QUEUE=QUEUE_NAME
MDL=$MDLAPPL$
```

The parameter statements must be coded in the form PARM=*VALUE* with no spaces on either side of the equal sign (=).

Parameter Settings

Set the parameter values as follows:

MDL	Specifies the pattern used to select the Execution Environment communications identifier for a batch server to use for communications.
QUEUE	The name of the queue you defined using the batch submission facility.
TDS	Identifies which Data Object Broker the batch server should connect to.
USERID	The name of the TIBCO Object Service Broker user ID used by this server to log in to the Data Object Broker. For example, you can use the name BATSRV1 and use BATSRV2, BATSRV3, and so on for subsequent servers.

Return Codes

The batch server can issue the following return codes:

Return Code	Meaning
0	Successful execution.
100	Missing or invalid startup parameters. The appropriate error message is printed in the job log listing.
200	Processing error. The appropriate error message is printed in the job log listing.

Chapter 15 TIBCO Object Service Broker Netview Support

This chapter describes how to use Netview with TIBCO Object Service Broker.

Topics

Overview

If your z/OS installation uses Netview, TIBCO Object Service Broker can connect to Netview and provide it with data to facilitate your operations management. This chapter discusses the extent of Netview support within TIBCO Object Service Broker, the information it provides to Netview, and the requirements for interoperation between TIBCO Object Service Broker and Netview.

TIBCO Object Service Broker Operational Messages

To effectively manage your network of computers, it is important to monitor critical events, expected or unexpected, that occur within the subsystems operating within the network. The Data Object Broker is one of those subsystems that supports the management of your network operations by keeping you informed of key events occurring within it. TIBCO Object Service Broker provides the following levels of information in the operational messages it creates: *log*, *informational*, *warning*, *error*, and *alert*.

Alert Messages

The most severe operational messages are alert messages. They indicate a major function failure or serious error condition (for example, a redolog failure). Most alert messages, such as those that pertain to error situations, require intervention because they indicate occurrences that impede normal operation of the Data Object Broker. The sooner an alert condition is corrected, the less impact it has on users.

Accessing Alert Messages

Alert messages are written to the SYSLOG with all other messages and are saved on a special *ALERT* page in the Pagestore, which can be viewed using option W from the Administration menu. Refer to W. Alert Messages on page 357 for more information about using this option. In addition, TIBCO Object Service Broker optionally supports Netview by sending alert messages to Netview to be highlighted for your operations staff.

Information Available on Netview

Netview alerts quickly communicate information to operations staff. TIBCO Object Service Broker supplies its messages to Netview, which displays them in a series of screens, described below.

Netview Screens

The following table contains a brief overview of Netview services used to present TIBCO Object Service Broker alerts. Refer to the appropriate Netview documentation for a complete description of operations and use.

Netview Screen	Function
Alert	Displays a one-line message identifying the resource and situation. An alarm is also sounded to attract the operator's attention.
Action	Appears when the operator selects the alert message. It identifies the job experiencing the condition and indicates one or two recommended actions.
Detail	Shows more information pertaining to the error reported: Job name and time the condition is detected. Type of event: notification, permanent loss of function, or temporary loss of function. A line of explanation on the condition specific to TIBCO Object Service Broker. Probable cause of the condition: The TIBCO Object Service Broker message code and message text.
Product	Displays information that describes the application where the error occurred, including release and modification data.

Activating Netview Support

Requirements

To activate TIBCO Object Service Broker Netview support, complete the following steps.

Step 1: Install Netview

Ensure that Netview is installed on the system.

Step 2: Make Netview Available to TIBCO Object Service Broker

The Netview runtime libraries must be available to TIBCO Object Service Broker, either via concatenation in the STEPLIB or in the link list. Contact your systems programer and review Netview requirements for your installation.

Step 3: Set Netview User-Defined Code Points

Netview User-Defined Code Points

The Netview User Defined Code Points are code values that IBM set aside for user applications. The range of codes is from x'E000' (57344) to x'EFFF' (61439). TIBCO Object Service Broker uses user codes for the Alert Descriptor and Probable Cause code points.

To support current and future TIBCO Object Service Broker needs, 32 consecutive code points should be reserved.

Setting the Base Code Point

For ease of installation, the base code point for the descriptor and cause are the same. If no other application uses user code points, use x'E000' as the base code point. The base code point indicator is passed in decimal format to TIBCO Object Service Broker during startup in the NETVIEWBASE parameter. The valid range of NETVIEWBASE is 0 through 16.

The formula used to calculate the actual base point is (value-1) * x'0100' + x'E000'. If the resulting value is zero, Netview support is not enabled.

Required User Code Points

The following are the user code points that must be added to the Netview tables. In each line, *xx* should be replaced to correspond to the formula described for NETVIEWBASE. If specified as 1, use x'E0'; if 5, use x'E4'.

In user defined code point table BNJ92UTB:

```
xx00 N HURON NOTIFICATION;
xx01 N HURON OPERATOR ALERT;
xx02 N HURON COMMITS SUSPENDED;
```

In user defined code point table BNJ93UTB:

```
xx01 N JOURNALS FULL;
xx02 N JOURNALS SPINNING;
xx03 N JOURNALS SPUN
xx04 N JRNL SPIN DEFERRED;
xx05 N JOURNAL FAILURE;
xx06 N THRESHOLD LIMITS;
xx07 N ACTIVATED
xx08 N SHUTDOWN
xx09 N ABNORMALLY ENDED;
xx0A N ABEND RECOVERY;
xx0B N I/O FAILURE;
```

Step 4: Set NETVIEWBASE Parameter

The NETVIEWBASE parameter must be set in the Data Object Broker's startup parameter list.

See Also The IBM Tivoli NetView documentation for more information on code points.

TIBCO Object Service Broker Parameters for information about the NETVIEWBASE Data Object Broker parameter.

Chapter 16 Operator Commands

This chapter lists TIBCO Object Service Broker operator commands.

Topics

Commands to Control a Data Object Broker

The format for Data Object Broker operator commands is:

```
MODIFY dob_jobname,osbcommand
```

or

```
F dob_jobname,osbcommand
```

where

`MODIFY or F`	The z/OS operator command.
dob_jobname	The name of your Data Object Broker batch job or the system task addressed by this command.
osbcommand	The TIBCO Object Service Broker operator command. Keywords and all commands except the TIBCO Object Service Broker **`Shutdown`** command can be abbreviated to a unique truncated form.

Issuing Data Object Broker Operator Commands

If you do not have access to a z/OS operator console, you can issue a Data Object Broker command in batch mode using S6BTLCMD, the Batch Operator Command utility. TIBCO Object Service Broker operator commands can also be issued using the Operator Functions option of the Administration menu, described in 2. Operator Functions on page 363.

See Also *TIBCO Object Service Broker for z/OS Utilities* for information about S6BTLCMD.

Commands

The following table describes the Data Object Broker operator commands and gives examples of how the commands are entered.

Command	Explanation	Example
BWOSTATUS	Displays the current Backup While Open (BWO) status of the page and journal DOB data sets. Refer to message S6BKF051I for an explanation of the information displayed. See related **FREEZE** and **UNFREEZE** commands. Refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for information about BWO.	**MODIFY** *dob_jobname*, **BWOSTATUS**

	Sample Output:
	`S6BKF051I- CURRENT BWO STATUS 100 BWO (00-00) S6B.EL.A50.PAGE1`
	`S6BKF051I- CURRENT BWO STATUS 100 BWO (00-00) S6B.EL.A50.PAGE2`
	`S6BKF051I- CURRENT BWO STATUS 100 BWO (00-00) S6B.EL.A50.PAGE3`
	`S6BKF051I- CURRENT BWO STATUS 100 BWO (00-00) S6B.EL.A5.JRNL2`
	`S6BKF051I- CURRENT BWO STATUS 100 BWO (00-00) S6B.EL.A5.JRNL1`	

Canceltermid=*termid* [a]	Quiesce a user connection using the supplied terminal ID. By quiescing by terminal ID you can differentiate between sessions.	**MODIFY** *dob_jobname*, **Canceltermid**=HNMAD001
Canceluser=*userid* [a, b] **Canceluser**=*userid;commnum*	Quiesce a user connection using the supplied User ID.	**MODIFY** *dob_jobname*, **Canceluser**=USR001
Checkpoint	Forces an immediate checkpoint.	**MODIFY** *dob_jobname*, **Checkpoint**
Comrestart	Requests the communications subsystem to restart any of the protocols that are currently inactive.	**MODIFY** *dob_jobname*, **Comrestart**

CRstatus	Queries the status of Data Object Broker tasks.
	Indicates the CPU time and elapsed time for Data Object Broker tasks.
	A single or double asterisk (*) beside a task indicates a potentially serious problem that should be investigated.
		MODIFY *dob_jobname*,**CRstatus** displays:
		`Task Status MM:SS SEC`
		` 0 Ready 00:13.68 .586`
		` 1 Ready 00:10.43 .498`
		` 2 Ready 00:11.22 1.275`
		` 3 Active 00:14.10 1.012`
		` 4 Ready 00:11.08 0.321`
		` 5 Active 00:46.22 1.210`
		`17965 I/O 11.064 SRB 110.817 TCB`

Command	Explanation	Example
CS,DISABLE	Attempts to disable processing via the HCS relay using the currently loaded parameters.	F Mycomponent,**CS,DISABLE**
CS,ENABLE	Attempts to enable processing via the HCS relay using the currently loaded parameters.	F Mycomponent,**CS,ENABLE**
CS,PARMDSN=	Modifies the name held by the RELAY for the RELAY parameter file. It does not load it. In order to load it use either the REFRESH command if this is sufficient, or use REFRESH followed by DISABLE and ENABLE the RELAY.	F Mycomponent,**CS,PARMDSN=**
CS,PARMDSN=*new.dataset.name*	Changes the parameter file to data set *new.dataset.name*.	F Mycomponent,**CS,PARMDSN=***new.dataset.name*
CS,PARMDSN=*new.dataset.name(xxxxxxxx)*	Changes the parameter file to member *xxxxxxxx* of data set *new.dataset.name*.	F Mycomponent,**CS,PARMDSN=***new.dataset.name(xxxxxxxx)*
CS,REFRESH	Reloads the RELAY parameter file that was allocated to the component at initialization time, or after modification by a subsequent F,Mycomponent,CS,PARMDSN= command. If an error occurs the RELAY continues processing without modifying the parameters in use. This allows an administrator to change the mapping of communication identifiers to TCP/IP hosts and ports. Changes to tcpipparms will not be honored and you will need to DISABLE and ENABLE the RELAY for any such changes to become effective.	F Mycomponent,**CS,REFRESH**
CS,STATUS	Displays the status of the RELAY and the name of the current RELAY parameter data set.	F Mycomponent,**CS,STATUS**
Dbjrnloff=*segname* or *segnumber*	Turns journal processing off for a specified segment. The segment referenced by this command must currently be online.	**MODIFY** *dob_jobname*, **Dbjrnloff**=sales or **MODIFY** *dob_jobname*, **Dbjrnloff**=1

Command	Explanation	Example
Dbjrnlon=*segname* or *segnumber*	Turns journal processing on for a specified segment. The segment referenced by this command must currently be offline. **Note** If journal processing is turned off in the Data Object Broker startup parameters, journals cannot be individually activated.	**MODIFY** *dob_jobname*, **Dbjrnlon**=sales
Dboffline=*segname* or *segnumber*	Varies the specified segment offline. To be varied offline, the segment must not be designated as a system segment in the DBDLIB. If there are pending updates for the specified segment, the segment is quiesced and is not terminated until the next checkpoint is complete.	**MODIFY** *dob_jobname*, **Dboffline**=sales
Dbonline=*segname* or *segnumber*	Varies the specified segment online.	**MODIFY** *dob_jobname*, **Dbonline**=sales
Dbreadonly=*segname* or *segnumber*	Sets the specified segment to read-only mode.	**MODIFY** *dob_jobname*, **Dbreadonly**=sales
Dbreadwrite=*segname* or *segnumber*	Sets the specified segment to accept updates, that is, normal mode.	**MODIFY** *dob_jobname*, **Dbreadwrite**=sales
Dbsegmentstatus	Displays the online/offline status of Pagestore segments.	**MODIFY** *dob_jobname*, **Dbsegmentstatus** displays:

```
A ID  NAME      D/S  MODE    JRN      PAGES      USED      READ     WRITES
O 000 CR2.SEG0    3  R/W-S   YES      54000     24432         0          0
O 001 CR2.SEG1    3  R/W      NO      54000      3114         0          0
F 004 CR2.SEG4    3  R/W      NO          0         0         0          0
```

Command	Explanation	Example
Forceuser=*userid* [b] **Forceuser**=*userid*; *commnum*	Disconnects a user ID from the Data Object Broker. **WARNING** Every effort is taken to ensure a clean termination but data integrity could be compromised. Use this command with extreme caution. The command releases all resources and logical locks, and frees the specified commarea, no matter the state of the commarea. If the command is issued and there was no previous CANCELUSER, the request is automatically downgraded to a CANCELUSER and a message is produced. The command is logged as an alert to ensure that there is a record in case of inconsistent results.	**MODIFY** *dob_jobname*, **Forceuser**=USR001
FREEZE	The FREEZE command flushes all updated pages from storage to the page data sets. If necessary, checkpoints are taken and flushed. When all the I/O activity for this operation is complete, the operator message S6BKX09I appears. While the TIBCO Object Service Broker system is frozen, transactions proceed as normal, but no write I/O is performed to the database and journals. If the BWO feature is used, the page and journal data sets that are under the control of DFSMSDFP are set into BWO status and enables the data sets for DFSMSDSS dump/copy operations. See also **BWOSTATUS** and **UNFREEZE**. Refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for information about BWO.	**MODIFY** *dob_jobname*, **FREEZE**
GTFstatus	Tells you whether a GTF task is active for collecting GTF records. See also related Data Object Broker parameters SSTRACE and MSGTRACE.	**MODIFY** *dob_jobname*, **GTFstatus**
Journaloff=*journalnumber* **Jrnloff**=*journalnumber*	Brings a journal data set offline, withdrawing it from the pool of journal data sets.	

Command	Explanation	Example
Journalon=*journalnumber* **Jrnlon**=*journalnumber*	Brings a journal data set online to make it available in the pool of journal data sets.	**MODIFY** *dob_jobname*, **Jrnlon**=01
Journalstatus **Jrnlstatus**	Displays the status and flags of the journal data sets. The statistics for the active journal (the one showing status flag A) include a percentage that indicates how full the journal is.	**MODIFY** *dob_jobname*, **Journalstatus**
MEMBERLIST	Displays the current status of all members of a Data Object Broker XCF group.	**MODIFY** *dob_jobname*, **MEMBERLIST**
Netrestart	Enables the TIBCO Object Service Broker network and cross-memory communications access, if possible.	**MODIFY** *dob_jobname*, **Netrestart**
Notrace=*userid* [b] **Notrace**=*userid*; *commnum*	Turns off specific user GTF tracing established by the command: **Trace**=*userid*	**MODIFY** *dob_jobname*,**Notrace**=usr01
Parm=*parmname*= *value* **Parameter**=*parmname*= *value*	Provides you with the ability to override many Data Object Broker startup parameters.	**MODIFY** *dob_jobname*, **Parm**=Chpagelimit=500
Peercancel=*applid*	Disconnects all inbound peer connections from the specified peer server. Refer to Chapter 13, Managing TIBCO Object Service Broker Networked Resources, on page 225 for more information about peer connections. **WARNING** The command releases all resources and logical locks, and frees the specified commarea, no matter the state of the commarea. Use the Peercancel command with extreme caution, only, for example, when a network problem has disrupted communications with the remote peer.	**MODIFY** *dob_jobname*, **Peercancel**=HKXU0003

Command	Explanation	Example
Quiesce	Sets the system to reject update requests from users. After the command is issued, the Administration menu (S6BTLADM) remains fully functional and new users cannot log in and existing users can browse only. Refer to Administration Menu on page 288 for more information.	**MODIFY** *dob_jobname*, **Quiesce**
Resume	Cancels the QUIESCE state, if set.	**MODIFY** *dob_jobname*, **Resume**
Shutdown	Shuts down the Data Object Broker. **Note** Shutdown cannot be abbreviated.	**MODIFY** *dob_jobname*, **Shutdown**
Spinenable=Yes or No	Sets or resets the DEFRSPIN flag if appropriate.	**MODIFY** *dob_jobname*, **Spinenable**=Y
Spinloadjcl=*nn*	Refreshes spin JCL in memory for the specified journal number without requiring that the Data Object Broker be restarted.	**MODIFY** *dob_jobname*, **Spinloadjcl**=1
Spinsubmit=Normal or Checkpoint or Immediate	Triggers a SPIN job submission according to the parameter specified: C (Checkpoint) Triggers a SPIN job at the next checkpoint. I (Immediate) Submits a SPIN job immediately. N (Normal) Cancels a SPINSUBMIT=Check-point request.	**MODIFY** *dob_jobname*, **Spinsubmit**=I
Startpeer=*applid*	Starts a specified outbound peer connection (for example, after a network outage). Before the request is processed, it is validated to ensure that the specified peer is valid and is not currently involved in connection processing. The number of peer connections must be below defined limits. Refer to Chapter 13, Managing TIBCO Object Service Broker Networked Resources, on page 225 for more information about peer connections and to the **Peercancel** command.	**MODIFY** *dob_jobname*, **Startpeer**=HKXU0003

Command	Explanation	Example
Stopserver= one of: *connection_id* ALLDBMS ALLHURON ALLREMOTE ALL*server_type* SRVID*group*	Quiesces the specified servers. If the server is currently associated with a transaction, it is not stopped until the transaction ends. *connection_id* is the unique identifier for the connection. This identifier typically consists of a prefix and a sequence number. It appears in the login message and also on the Resource Manager PATH LIST screen of the Administration menu. Refer to 3. Resource Management on page 364 for more information. ALLDBMS stops all external database servers. Outbound peer servers not affected. ALLHURON stops all outbound peer connections. Same as ALLHRN. ALLREMOTE combines ALLDBMS and ALLHURON. ALL*server_type* stops all servers of *server_type*, which is a three-character abbreviation for the type of server; for example, Adabas servers have a type of ADA, peer servers (also referred to as rules API servers) have type API. SRVID*serverid* stops all servers with a matching *serverid*.	**MODIFY** *dob_jobname*, **Stopserver**=IMS01 **MODIFY** *dob_jobname*, **Stopserver**=ALLDB2 **MODIFY** *dob_jobname*, **Stopserver=**SRVIDDEFAULT0

Command	Explanation	Example
Sweepaction=*(action, userid)* [b] **Sweepaction**=*(action, userid;commnum)*	Modifies the previously specified method of Page Sweep reporting for the specified user ID. This method is specified by either a Sweepaction command or by the PAGESWEEPACTION Data Object Broker parameter. Valid *action* values: DUMP: An SVC dump is taken NODUMP: Disable the request for SVC dumps for page sweep violations LOG: A warning message appears in the log and on the console NOLOG: Disable logging of the warning message SMF: An SMF record is written NOSMF: Disable the creation of SMF records for page sweep violations *userid* is the user ID of the connection. *commnum* is required only in the case of duplicate user IDs.	**MODIFY** *dob_jobname*, Sweepaction=(SMF,usr40;1) **MODIFY** *dob_jobname*, Sweepaction=(NODUMP,usr40)
Sweeplimit=*(limit, userid)* [b] **Sweeplimit**=*(limit, userid;commnum)*	Modifies the previously specified threshold at which to trigger Page Sweep reporting. This limit is specified by either a Sweeplimit command or by the PAGESWEEPLIMIT Data Object Broker parameter. Valid values are 0 to 65535. A value of 0 disables Page Sweep reporting for the specified user. *userid* is the user ID of the connection. *commnum* is required only in the case of duplicate user IDs.	**MODIFY** *dob_jobname*, Sweeplimit=(1000,usr40)

Command	Explanation	Example
SWITCH= one of: MEMBERNAME ANY FORCE ALLOW	Controls switch processing. MEMBERNAME and ANY are only supported by the primary Data Object Broker. FORCE and ALLOW are only supported by a secondary Data Object Broker. MEMBERNAME – terminate the current primary Data Object Broker and switch to the secondary Data Object Broker identified by MEMBERNAME. ANY – terminate the current primary Data Object Broker and switch to any available secondary Data Object Broker. FORCE – if there is no active primary Data Object Broker in the Data Object Broker XCF group, initialize this secondary Data Object Broker to become the primary Data Object Broker. ALLOW – allow this secondary Data Object Broker to be the target of a SWITCH command (if the secondary Data Object Broker is set to the NSWITCH state due to a mismatch of configuration data with the primary Data Object Broker, then ALLOW will set this secondary Data Object Broker's state to 2READY, thus allowing it to be the target of a SWITCH command).	**MODIFY** *dob_jobname*, **SWITCH**=ALLOW
Timestamp	Issues a TIBCO Object Service Broker date/time WTO message to the system log.	**MODIFY** *dob_jobname*, **Timestamp**
Traceid=*userid* [b] **Traceid**=*userid*; *commnum*	Traces all system service activity for the specified *userid*. Data is written to GTF. Appropriate GTF setup is required. **Note** You can either use the full system service trace (Sstrace), or you set the trace for a specific *userid*.	**MODIFY** *dob_jobname*, **Traceid** =USR08

Command	Explanation	Example
UNFREEZE	The UNFREEZE command allows I/O activity to database and journals to resume. If BWO support is used, prior to resuming I/O, the page and journal data sets that are under the control of DFSMSDFP are reset to normal access thus disabling concurrent access. See also **BWOSTATUS** and **FREEZE**. Refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for information about BWO.	**MODIFY** *dob_jobname*, **UNFREEZE**
Usercount	Displays a count of TIBCO Object Service Broker user IDs by category (for example, users, external database servers, jobs). See also **Userlist**.	**MODIFY** *dob_jobname*, **Usercount**

Command	Explanation	Example
Userlist	Displays a list of all users who are logged in, followed by the counts in each category (for example, users, external database servers, jobs). If no users are logged in, only user counts appear. See also **Usercount**.	**MODIFY** *dob_jobname*, **Userlist** The list of users is given in the format x-uuuuuuuu where the x is the session type prefix, followed by a - or + character, and then a character user identifier uuuuuuuu. The second character indicates whether the session is being traced or not. + indicates tracing for the session is active; - indicates no tracing. See command TRACEID for more information on tracing. The type of session is identified by the one character prefix. The meanings are: " " (blank) – TSO user C – CICS user D – DBMS or rules server session J – Batch job N – Native Execution user U – UNIX user I – IMS/DC user L – deferred logoff O – Operator or OAI session t – Standby / "other type" session P – Peer server (HIN)
Usermax=*value*	Adjusts the maximum number of Execution Environment users allowed to log in to TIBCO Object Service Broker. Users already logged in and administrators are not affected. Minimum value is 0. Maximum is the same as the Maxuser value defined at TIBCO Object Service Broker startup. **Usermax**=0 can be useful to prevent anyone logging in just before a Data Object Broker shutdown.	**MODIFY** *dob_jobname*, **Usermax**=0

Command	Explanation	Example
Wtoprint	Closes the TIBCO Object Service Broker WTO message log SYSOUT file. Open a new WTO log file.	**MODIFY** *dob_jobname*, **Wtoprint**
Wtosysout=*value*	Determines whether TIBCO Object Service Broker WTO messages are written to a SYSOUT file.	**MODIFY** *dob_jobname*, **Wtosysout**=yes

a. If the connection is idle, that is, waiting for an Execution Environment request, it is terminated immediately. If the Data Object Broker is processing a query on behalf of the Execution Environment, the connection is terminated when the next service request is received by the query processor. If the Data Object Broker is processing a commit, the connection is terminated when the commit is completed.

b. If the specified user ID is not unique, a list of connections with the specified user ID appears. This list contains additional information to help identify the connection to be processed. You must reenter the command with ";commnum" where commnum is selected from the list.

Commands to Modify Data Object Broker Startup Parameters

The format for commands to modify Data Object Broker startup parameters is:

```
MODIFY dob_jobname,Parm=parm_name=value
```

where

MODIFY or **F**	The z/OS operator command.
dob_jobname	The name of the batch job or the system task name under which the Data Object Broker is running.
Parm=	The TIBCO Object Service Broker operator command for changing a startup parameter.
parm_name	The name of the startup parameter you want to override.
value	The new value that you want to assign to the parameter.

For example, to set the SMF record number to 217 and start Data Object Broker SMF data collection:

```
MODIFY S6ELDOBA,PARM=SMFRECORD=217
```

Dynamically Changing Data Object Broker Parameters

Parameters, usually set at installation time, determine operating characteristics of a Data Object Broker. And while a Data Object Broker is up and running, you can change the values or settings of most of these parameters. You are effectively altering the Data Object Broker's operating characteristics dynamically.

Modification Categories

There are four modification categories for the parameters:

- Parameters that cannot be modified, such as MAXUSERS.
- Those that can be modified once only, that is, SMFRECORD and WTOSUFFIX.
- Those that can be modified any number of times, such DBSNAP.

- Those that can be modified but it is not recommended.

When you modify a Data Object Broker parameter that is an interval (such as CHPTINTERVAL and SPININTERVAL), TIBCO Object Service Broker immediately ends any active interval of the corresponding type and starts a new interval using the new value.

See Also *TIBCO Object Service Broker Parameters* for detail about each of the Data Object Broker parameters.

Commands to Control the Native Execution Environment

The format for TIBCO Object Service Broker Native Execution Environment operator commands is:

```
MODIFY ee_jobname,osbcommand
```

or

```
F ee_jobname,osbcommand
```

where

MODIFY or F	The z/OS operator command.
ee_jobname	The name of the Native Execution Environment batch job or the system task addressed by this command.
osbcommand	The operator command to the Execution Environment.

Commands

The following table lists the Native Execution Environment operator commands, describes them, and gives examples of how the commands are entered.

Command	Explanation	Example
Canceluser= *userid* or *termid*	Disconnects a user ID or terminal ID from the Execution Environment specified by jobname **Canceluser** is ignored while ESTAE recovery is in progress. **WARNING** If multiple sessions can be associated with the same user ID, specify the terminal ID of the session otherwise an incorrect session could be canceled.	**MODIFY** *ee_jobname*, **Canceluser**= USR001
CS,DISABLE	Attempts to disable processing via the HCS relay using the currently loaded parameters.	**F** *ee_jobname*,**CS,DISABLE**
CS,ENABLE	Attempts to enable processing via the HCS relay using the currently loaded parameters.	**F** *ee_jobname*,**CS,ENABLE**
CS,PARMDSN=*new.dataset.name*	Changes the parameter file to data set *new.dataset.name*.	**F** *ee_jobname*,**CS,PARMDSN**= *new.dataset.name*

Command	Explanation	Example
CS , PARMDSN=*new.dataset .name(xxxxxxxx)*	Changes the parameter file to member *xxxxxxxx* of data set *new.dataset.name*.	**F** *ee_jobname*,**CS , PARMDSN**= *new.dataset.name(xxxxxxxx)*
CS , REFRESH	Reloads the RELAY parameter file that was allocated to the component at initialization time, or after modification by a subsequent F,Mycomponent,CS,PARMDSN= command. If an error occurs the RELAY continues processing without modifying the parameters in use. This allows an administrator to change the mapping of communication identifiers to TCP/IP hosts and ports. Changes to tcpipparms will not be honored and you will need to DISABLE and ENABLE the RELAY for any such changes to become effective.	**F** *ee_jobname*,**CS , REFRESH**
CS , STATUS	Displays the status of the RELAY and the name of the current RELAY parameter data set.	**F** *ee_jobname*,**CS , STATUS**
Setnumserver	Sets the maximum number of servers of the specified type that can run with this Execution Environment Valid types are: ADA for Adabas API for TIBCO Object Service Broker peer servers DAT for CADatacom IMS for IMS/DB IMSCICS for CICS DL/I servers STB for standby sessions You can specify up to a maximum of 128 servers. For CICS DL/I servers, the maximum must be less than or equal to the TASKEXEC parameter.	**MODIFY** *ee_jobname*, **Setnumserver**=32, TYPE=API

Command	Explanation	Example
Shutdown or **Shut**	Shuts down the Execution Environment as soon as currently logged in users are logged out, while disabling new logins. Long-running sessions, such as Object Integration Gateway, SDK (Java), or inactive logged in users could cause the Execution Environment to wait indefinitely. Use the **Shutimmed** command to close these after current users are logged out. **Note** You can also enter the following z/OS STOP command to get similar results: `P ee_jobname`	**MODIFY** *ee_jobname*, **Shutdown**
Shutimmed or **Shuti**	Causes the Execution Environment to immediately shut itself down. As a rule, first issue the **Shutdown** command and wait for active users to log out before issuing this command.	**MODIFY** *ee_jobname*, **Shuti**
Startnumserver	Starts the given number of the specified type of server. Valid types are: ADA for Adabas API for TIBCO Object Service Broker peer servers DAT for CADatacom IMS for IMS/DB IMSCICS for CICS DL/I servers STB for standby sessions You can specify 1 to 128 servers to start, subject to the maximum number of servers allowed for the particular type. Refer to the command **Setnumserver** for more information.	**MODIFY** *ee_jobname*, **Startnumserver**= 3, TYPE=DAT

Command	Explanation	Example
Stopnumserver	Stops the given number of the specified type of server. Valid types are: ADA for Adabas API for TIBCO Object Service Broker peer servers DAT for CADatacom IMS for IMS/DB IMSCICS for CICS DL/I servers ORS for Oracle SLK for the Gateway for ODBC STB for standby sessions Refer to the command **Setnumserver** for more information.	**MODIFY** *ee_jobname*, **Stopnumserver**=2, TYPE=IMS
Userlist	Displays information about logged on users	**MODIFY** *ee_jobname*,USERLIST

Commands to Control the CICS Execution Environment

The format for TIBCO Object Service Broker CICS operator commands is:

```
MODIFY cics_jobname,osbcommand
```

or

```
F cics_jobname,osbcommand
```

where

MODIFY or **F**	The z/OS operator command.
cics_jobname	The name of the CICS Execution Environment batch job or the system task addressed by this command.
osbcommand	The operator command to the Execution Environment.

Commands

The following table lists the TIBCO Object Service Broker CICS operator commands, describes them, and gives examples of how the commands are entered:

Command	Explanation	Examples
HINT	Starts or restarts a CICS Execution Environment and establishes communications with a Data Object Broker. You can specify the following optional parameters for **HINT**: Mdl = (Model) TDS= (Table Data Store) Instlib=Installation Library Syslib=System Library	**MODIFY** *cics_jobname*, **HINT** TDS=EP01SRV, INSTLIB=BANK03
HINQ	States whether the specified TIBCO Object Service Broker CICS Execution Environment is attached or detached and lists the number of users, their user IDs, and termids You can specify **HINQ USERC** to suppress the listing of users and get only the user count or active sessions.	**MODIFY** *cics_jobname*, **HINQ USERC**

Command	Explanation	Examples
HCUS	Cancels the user session specified by the *userid/termid*. This is the normal method to cancel a user session and should be used before the **HFUS** transaction described below. For diagnostic information about the user session, specify **HCUS** *userid*,Dump or **HCUS** *termid*,Dump.	**MODIFY** *cics_jobname*, **HCUS** *userid* or *termid*
HFUS	Forces the cancellation of the user session specified by the *userid/termid* **WARNING** Use with caution and only when the need to cancel a user session overrides the potential for unpredictable results. Consider restricting access to this Transaction by authorized personnel only by implementing transaction security.	**MODIFY** *cics_jobname*, **HFUS** *userid* or *termid*
HREL	Used to display the status of and control the TCP/IP RELAY communications component of TIBCO Object Service Broker within the CICS job. This component operates independently of any CICS terminal control HCS facility. The RELAY is useful to connect an Eclipse or remote client to the CICS execution environment. For more information, see the description of the corresponding **CS,xxxxxxx** RELAY commands in Commands to Control a Data Object Broker. Note that in CICS, the RELAY commands are invoked by the **HREL** transaction, followed by a space and the RELAY sub-command. The CS, command prefix used for other job environments is not used. The **HREL** transaction is designed for use by the CICS z/OS console operator. You may enter the **HREL** transaction at a CICS terminal, but the response will be displayed on the system console by a WTO.	Stop the RELAY: **MODIFY** *cics_jobname*, **HREL DISABLE** Start the RELAY: **MODIFY** *cics_jobname*, **HREL ENABLE** Change the parameter file of the RELAY: **MODIFY** *cics_jobname*, **HREL PARMDSN=** *new_dataset_name(member)* Reload parameter file definitions held in parameter file of RELAY: **MODIFY** *cics_jobname*, **HREL REFRESH** Display status of the RELAY: **MODIFY** *cics_jobname*, **HREL STATUS**
HTRM	Shuts down the TIBCO Object Service Broker CICS Execution Environment without shutting down CICS You must specify one of the following operands: **SHUT** to shut down the Execution Environment when all currently logged in users are logged out. No new users can log in. **SHUTI** to immediately terminate the Execution Environment by canceling all logged in users.	**MODIFY** *cics_jobname*, **HTRM SHUT** **MODIFY** *cics_jobname*, **HTRM SHUTI**

Command	Explanation	Examples
HSRV	Sets the number of servers, or starts or stops a server You must specify one of the following operands: SETNUMSERVER=*nn*, TYPE=API or IMS Sets the maximum number of external database server sessions that can be started in this CICS environment for the specified server type. STARTNUMSERVER=*nn*, TYPE=API or IMS Starts the specified number of external database server sessions for the specified server type. Corresponding non-terminal CICS transactions as specified by the CICS HURONTRAN=*transid* Execution Environment parameter are started through the TIBCO Object Service Broker **MODIFY** commands via the CICS console CONSOLE(0) STOPNUMSERVER=*nn*,TYPE=API or IMS Stops the specified number of external database server sessions for the specified server type.	**MODIFY** *cics_jobname*,**HSRV** SETNUMSERVER=32, TYPE=API **MODIFY** *cics_jobname*,**HSRV** STARTNUMSERVER=16, TYPE=IMS **MODIFY** *cics_jobname*,**HSRV** STOPNUMSERVER=16, TYPE=API

Chapter 17 Using the Interface to TIBCO Mainframe Service Tracker

This chapter describes how to use the interface to TIBCO Mainframe Service Tracker.

Topics

Overview

In addition to the monitoring available with TIBCO Object Service Broker using the Administration Menu, an interface to TIBCO Mainframe Service Tracker is also available. Using this interface, a job or started task associated with a TIBCO Object Service Broker product is able to send a job initialization and job termination notification message to TIBCO Mainframe Service Tracker. This allows TIBCO Object Service Broker to take part in TIBCO Mainframe Service Tracker monitoring, and TIBCO Mainframe Service Tracker in turn feeds this information into TIBCO Hawk.

Enabling the Interface

To enable the TIBCO Mainframe Service Tracker interface, perform the following for each TIBCO Object Service Broker product job or started task that you wish to monitor.

1. Create a sequential data set that contains a single entry consisting of a `SSID=tracker_subsystem_id` parameter, starting in column one. The `tracker_subsystem_id` is the four character subsystem ID specified when the TIBCO Mainframe Service Tracker subsystem was installed; for example, `SSKS`.

   ```
   SSID=SSKS
   ```

2. Add a TRACKER DD statement to the job's JCL that specifies the name of the data set created in step 1 (in the example below, `your.ssidparm.dataset`).

   ```
   //TRACKER DD DISP=SHR,DSN=your.ssidparm.dataset
   ```

See Also

TIBCO Mainframe Service Tracker Installation and Administration for details on setting up and using TIBCO Mainframe Service Tracker, as well as details on how it consumes the data received from TIBCO Object Service Broker.

For details about using the Administration Menu to monitor TIBCO Object Service Broker, see Administration Menu, page 288.

TIBCO Hawk® documentation.

TIBCO Administrator™ documentation.

Notification Record Layout

The following table shows the layout of the Notification Record sent to TIBCO Mainframe Service Tracker:

Data Format	Description
CL10	Message ID (S6BHK100 or S6BHK200)
CL1	Message severity (I)
CL1	Filler (x'00')
CL8	Filler (blanks)
CL16	z/OS job name
CL64	TIBCO Object Service Broker product name
CL32	Filler (blanks)
CL64	z/OS host name
CL16	'O-S-B'
CL256	Message text

Chapter 18 Monitoring and Controlling the TIBCO Object Service Broker Environment

This chapter describes how to monitor and control the TIBCO Object Service Broker environment.

Topics

Overview

Displaying the Administration Menu

You can monitor and control your TIBCO Object Service Broker system through the TIBCO Object Service Broker Administration menu. To display the Administration menu and its associated screens, execute the S6BTLADM utility. The ADMIN EXEC supplied with TIBCO Object Service Broker provides an example of how to invoke the S6BTLADM utility.

Administration Menu

The Administration menu contains a header line and a selection of menu options.

```
S6BADM$1  S6DCDOBA       ADMINISTRATION MENU                2009JUN19 07:29:40

   STATISTICS                                DIAGNOSTIC DISPLAYS

   A  GENERAL                                P  DISPLAY RESIDENT PAGE BUFFER
   B  SEGMENT/DASD                           S  MONITORING PROCESS
   C  LOCK MANAGER                           T  MEMORY DISPLAY
   D  BUFFER POOLS                           U  PAGE IMAGE
                                             V  RESIDENT PAGE POOL DIRECTORY
   F  PROFILE DISPLAYS                       W  ALERT MESSAGES
   G  UPDATED PAGES PER TRANSACTION          Z  DIAGNOSTIC DUMPS
   H  PHYSICAL CPU AND I/O STATISTICS
   I  USER ACTIVITY                          ADMINISTRATION CONTROLS
   J  INSTALLATION/CONFIGURATION
                                             1  IN-DOUBT TRANSACTIONS
                                             2  OPERATOR FUNCTIONS
                                             3  RESOURCE MANAGEMENT


                         ENTER SELECTION: ____________________

   STANDARD FUNCTIONS:    PF1-HELP    PF3-MENU    PF6-SCREEN PRINT    PF12-EXIT
  WELCOME TO THE ADMINISTRATOR, PRESS PF1 ON ANY SCREEN FOR HELP
```

Header Line

The header line of the screens within the Administration menu provides information about the displayed screen.

The header line fields in the illustration are as follows:

S6BADM$1	The screen name.
S6DCDOBA	Name of the Data Object Broker that is hosting this operator session.
ADMINISTRATION MENU	The screen title.
2009JUN19 07:29:40	Date and time of the last screen refresh.

Navigating the Screens

From any screen in the Administration menu, you can use the following key commands:

PF1	Display help information about the current screen.
PF3	Return to the main Administration menu.
PF6	Print the current screen.
PF7	Scroll backwards where indicated.
PF8	Scroll forwards where indicated.
PF12	Exit from the Administration menu.

Controlling Access

TIBCO Object Service Broker always uses external security to control access to the Data Object Broker, including the use of the Administration menu. The z/OS security interface, System Authorization Facility (SAF), is used to verify accesses to the Data Object Broker. Use the SECURADMIN=Y Data Object Broker parameter to enable SAF.

See Also *TIBCO Object Service Broker Managing Security* for more information about using external security to secure the Administration menu

TIBCO Object Service Broker Parameters for a description of SECURADMIN

Categories of Administration Options

The Administration menu divides the various menu options into these categories:

Statistics	Displays statistics accumulated since TIBCO Object Service Broker last started.
Diagnostic Displays	Displays information to help you analyze your system and diagnose potential problems.
Administration Controls	Gives you administration control over the Data Object Broker and TIBCO Object Service Broker resources.

The following describes the Administration menu options available in these categories.

Statistical Menu Items

A. General

This section describes three general screens: General Statistics, Enclave Statistics, and TCB Statistics.

General Statistics Screen

Function Option A, GENERAL, displays the statistics accumulated since TIBCO Object Service Broker last started.

Example

```
S6BADMA1  S6DCDOBA              GENERAL STATISTICS            2011JAN06 07:51:23

 EXCEPTION STATISTICS
SENDERR                 0   SYNAD                    0   RPMLOCK                 0
GTF ERR                 0

 POTENTIAL DELAYS (MAY INDICATE A CONFIGURATION OR PROCESSING ISSUE)
WAITS                 410   CHPT-I                   0   FLUSH                   0
Q-RETRY                 0   QR FAIL                  0   WRAP                    0
DEF PG               8775   PG WAIT               1049   INDBTQE                 0
SNAPS                   0   DEADLCK                  0   EXCP-R                  0

 REQUEST HANDLING AND PROCESSING METRICS
SUPV                    2   CHPT                     0   COMM                  201
FILE                 7170   RTUQ                     3   APPL                282177
OPER                   12   CONS                     0   TIMR                    5
PH0                     0   PH1                 267678   PH2                  1182
SERVER                  0   INDOUBT                  0

    LAST CHPT 2011JAN05 09:42:24 CHPT#      926       COMPLETE      926
    LAST SPIN 2011JAN06 07:49    JRNL#        2       JRNL%           0

 PHYSICAL PROCESSING PERFORMED
READ                 5474   RULE                    47   CTAB                 3688
WRITE                   0   REPL                   262   CTB S/R                 0
DEL                  2289   INS                    674   CICS DISPLAY            0
JRNL                    0   SMF                      3   REDOLOG              2810
EXCP                    0   WTO                     60   GTF                     0
SEND                269051   RECEIVE             269056   HITHASH              5475
LOGONS                 11   LOGOFF                   2   MISHASH              5610
MAXUSR               1000   MAXCON                   9   TAMUSR                  4

 LOGICAL PROCESSING METRICS
GET                925994   RULE                 12632   CTAB                353515
PUT                 96904   REPL                     4   INS                   9786
```

```
DEL                 76047   SYNC                1182
GET4K                3668   FRE4K               2486
GETF                  135   PUTF                 857
LOCK               145207   UNLOCK               352   DEF LCK                0
GETBUF            1207122   FREEBUF          1206777   DEF BUF                0
GETXTAB                 0   FREEXTB                0
    ENTER-REFRESH TOTALS  PF4-TASK/TCB STATISTICS  PF5-ENCLAVE STATISTICS
NO MORE DATA TO BE DISPLAYED
```

Fields

This section defines the fields on the General Statistics screen, as follows:

- **Exception Statistics** — Exception statistics show error conditions that can be caused by processing failures within the TIBCO Object Service Broker environment. The values could be accompanied by errors in the system log.

SENDERR	The number of failed send requests detected by the Query task when attempting to send a response back to the user.
SYNAD	The number of VSAM I/O errors detected in both the Pagestore file handler and the checkpoint handler.
RPMLOCK	The number of locked pages with no owner. This is an internal processing error and should be reported to TIBCO Support.
GTF ERR	The number of General Trace Facility write requests that terminated with a non-zero return code.

- **Potential Delays** — The values of potential delays could denote configuration or environmental issues that are adversely affecting performance and throughput.

WAITS	The number of times a transaction was forced to wait for a logical lock or CTABLE resource. A logical lock wait occurs when another transaction holds the required lock, typically the result of transaction mix. A CTABLE is a table definition composite. A wait occurs when a CTABLE is being built by one transaction and a second transaction must wait for its completion. A high number of CTABLE waits could indicate that the Resident CTABLE Pool (CTABRESIDENT parameter) is set too small for the transaction mix.
CHPT-I	The number of page images copied because of checkpoint interference. Checkpoint interference is caused when a page within a checkpoint is required by a transaction outside the scope of the checkpoint. A copy of the page image is taken and used by the current transaction. Checkpoint interference is mainly the result of transaction mix.

FLUSH	The number of checkpoints flushed. During checkpoint processing, page updates are written to the cache. When the cache I/O is complete, the next checkpoint build starts. Page images are recorded to the journal and Pagestore while the next checkpoint is being built. To reduce calls and smooth out I/O, page image updates are piggy-backed on to read to the same data set. As the next checkpoint build reaches a threshold, pages from the previous checkpoint not propagated back onto the Pagestore are written. This is referred to as flushing the checkpoint. If the majority of the checkpoints have to be flushed, the checkpoint size could be too low.
Q-RETRY	The number of query transactions retried after a recoverable failure. The retrying of queries tends to result from environmental conditions that are transient, based on the current transaction mix. There is little that can be done to reduce the potential of these happening.
QR FAIL	The failed query retry count.
WRAP	The number of times the Resident Page Pool was interrogated and no slot was available that could be reassigned. A non-zero value in this field indicates that the Resident Page Pool is not large enough to accommodate the workload.
DEF PG	The deferred page count; indicates the number of times a transaction waited because a required page was physically held by another user's commit. A non-zero value indicates a delay in throughput that results from the transaction mix and it is not readily easy to retune.
PG WAIT	The page wait count; indicates the number of sessions put on queues waiting for a locked page.
INDBTQE	The number of in-doubt queries issued that could not be resolved (in-doubt query errors). When an external resource is interrupted during the processing of a Fail Safe level-1 or level-2 commit, the transaction is said to be in-doubt. When an appropriate resource becomes available, a query is sent to the resource to determine its status. If for some reason the resource cannot determine the status, the query fail error counter is incremented.
SNAPS	The number of snap dumps processed as a result of failed transactions.
DEADLCK	The number of transactions aborted because a deadlock situation is detected.
EXCP-R	The number of EXCP write retry errors; EXCP write errors to the contingency log and the cache that were retried.
	• **Request Handling and Processing Metrics**
SUPV	The number of invocations of the supervisor task.

CHPT	The number of invocations of the checkpoint task.
COMM	The number of invocations of the communications task.
FILE	The number of invocations of the file handling task.
APPL	The number of invocations of the application/query task.
OPER	The number of operator messages.
CONS	The number of modify messages received from the operator console and passed onto the operator task.
TIMR	The timer task event count.
PH0	The number of transactions recovered at startup.
PH1	The number of query transactions.
PH2	The number of commit transactions.
SERVER	The number of messages sent to external servers and peer Data Object Brokers.
INDOUBT	The number of uncommitted transactions held in-doubt.

- **Other Information**

LAST CHPT	The date and time of the last checkpoint.
CHPT#	The current checkpoint number.
COMPLETE	The number of the last completed checkpoint.
LAST SPIN	The date and time of the last journal spin.
JRNL#	The active journal number.
JRNL%	The percentage of the active journal that is full.

- **Physical Processing Performed**

READ	The number of pages read.
RULE	The number of rule pages read.

CTAB	The number of control table (table definition composite) requests where the CTABLE was built.
WRITE	The number of VSAM pages written to the Pagestore.
REPL	The number of pages updated with the content length unchanged.
CTB S/R	The number of CTABLEs built for screens and reports.
DEL	The number of page updates where the content length decreased.
INS	The number of page updates where the content length increased.
CICS DISPLAY	The number of display CICS connections.
JRNL	The journal file I/O count.
SMF	The System Monitoring Facility (SMF) statistics file I/O count.
REDOLOG	The number of intent lists written to the redolog data set.
EXCP	The number of EXCP writes, including those to the contingency log and the cache data sets.
WTO	The number of Write-To-Operator messages written.
GTF	The number of General Tracing Facility records written.
SEND	The number of messages sent to Execution Environments, servers, and peer Data Object Brokers.
RECEIVE	The number of messages received from Execution Environments, servers, and peer Data Object Brokers.
HITHASH	The number of collisions on the Resident Page Index hash chain.
LOGONS	The number of logins processed.
LOGOFF	The number of logouts processed.
MISHASH	The number of single entries on the Resident Page Index hash chain.
MAXUSR	The maximum number of online and batch user sessions allowed at one time, as specified by the MAXUSERS Data Object Broker parameter.

MAXCON	The high water mark for all connected sessions, including online and batch user sessions, operator, and external and peer server sessions.
TAMUSR	The current number of connected user online and batch sessions.

- **Logical Processing Metrics**

GET	The number of page read requests.
RULE	The number of rule page read requests.
CTAB	The number of control table read requests.
PUT	The number of logical page writes requested.
REPL	The number of rows replaced.
INS	The number of rows inserted.
DEL	The number of rows deleted.
SYNC	The number of commit requests serviced.
GET4K	The number of 4 KB work blocks requested.
FRE4K	The number of 4 KB work blocks released.
GETF	The number of free page requests.
PUTF	The number of pages released.
LOCK	The number of logical locks requested.
UNLOCK	The number of logical locks released.
DEF LCK	The number of logical lock requests deferred due to lock contention.
GETBUF	Number of pool buffers requested.
FREEBUF	Number of pool buffers released.
DEF BUF	The number of pool buffers requested that were deferred due to availability.
GETXTAB	The number of Xtable buffers requested.
FREEXTB	The number of Xtable buffers released.

- **Key Commands**

Enter	Refresh data.
PF1	Display help information about the current screen.
PF3	Return to the main Administration menu.
PF4	Task/TCB Statistics.
PF5	Enclave Statistics.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF12	Exit from the Administration menu.

Enclave Statistics Screen

Function The Enclave Statistics screen presents a summary of TIBCO Object Service Broker Enclave statistics.

Example

```
S6BADMA3  S6DCDOBA       ENCLAVE STATISTICS             2011JAN06 07:52:05

                                TOTAL              DELTA
                             HHH MM SS.MICRO    HHH MM SS.MICRO
    ZIIP ON CP TIME          000 00 00.769447   000 00 00.050681
    ZIIP QUALIFIED TIME      000 02 28.038771   000 00 10.457387
    ZIIP CPU TIME            000 02 27.269324   000 00 10.406706
    CPU TIME                 000 02 28.038843   000 00 10.457459

    PHASE 1 REQUESTS                  561452             31749
    PHASE 1 SRB REQUESTS              561452             31749


                ENTER-REFRESH TOTALS  PF5-GENERAL STATISTICS
```

Information Displayed

For each value, a total for this instance of the Data Object Broker and a delta value between each refresh of this screen are displayed. CPU times are normalized, that is, the CPU times given for zIIP processors are those that would have been given for a regular processor to run the same work. ZIIP QUALIFIED work is work that is eligible to be executed on a zIIP processor regardless of where it was actually executed.

See the following table for details.

Column	Description
ZIIP ON CP TIME	The CPU Time zIIP-qualified work used on a regular processor
ZIIP QUALIFIED TIME	The CPU time for zIIP-qualified work
ZIIP CPU TIME	The CPU time used on a zIIP processor
PHASE 1 REQUESTS	The number of Phase 1 requests
PHASE 1 SRB REQUESTS	The number of Phase 1 requests dispatched in SRB mode

TCB Statistics Screen

Function The TCB Statistics screen presents a summary of TIBCO Object Service Broker TCB usage statistics. The screen is present for all active TCBs.

Example

```
S6BADMA2  S6H1DOBB            TCB STATISTICS                  2006NOV29 09:43:50

  TASK  STATUS      TIME BUSY      TCB TIME   MSGS PROCESSED INFLIGHT  MAX-CON
   0    READY            0:06         0.214
   1    READY            0:01         0.190
   j    READY            0:00         0.006
   2    ACTIVE           0:09         2.120
   3    READY            0:03         0.803              572        0        1
   31   READY            0:00         0.406
   4                                  20.628
    1   READY            0:26        20.628           121601        0        2
   6    READY            0:00         0.040
   c    READY            0:00         0.007
   t    READY            0:03         3.516

                ENTER-REFRESH TOTALS  PF4-GENERAL STATISTICS
```

Information Displayed

Column	Function
TASK	The task number.
STATUS	The current status of the TCB: Ready – idle and ready to process work Active – processing work Ended – the TCB ended /Shut – suffix meaning shutdown in progress S*nn*U*nn* – abnormal termination code
TIME BUSY	The total amount of time the TCB was processing, reported in elapsed time.
TCB TIME	The total amount of CPU time used by the TCB.

For the Query (4) and Sync (3) tasks, three additional statistics appear. The Query Task TCB (4) has a summary line and then a line for each task running. This format is used to support the Multiple Query Task feature.

Column	Function
MSGS PROCESSED	The total number of messages assigned to the specific TCB.
INFLIGHT	The number of messages currently assigned to the specific TCB.
MAX-CON	The maximum number of messages assigned to the TCB at one time.

Key Commands

Enter	Refresh data.
PF1	Display help information about the current screen.
PF3	Return to the main Administration menu.
PF4	Display the General Statistics screen.
PF6	Print the current screen.
PF12	Exit from the Administration menu.

B. Segment/DASD

Function Option B, SEGMENT/DASD, displays segment and DASD statistics.

Displaying Information for All Segments or for a Specified Segment

You can display segment and DASD statistics for all segments, or for a specified segment. From the ENTER SELECTION prompt on the ADMINISTRATION MENU screen, type one of the following:

B	To display the SEGMENT STATISTICS screen, which displays all segments.
B*seg*	Where *seg* is the number of an active segment. This takes you directly to the DASD STATISTICS screen for the specified segment.

If you attempt to go directly to the DASD STATISTICS screen with a segment that is offline or invalid, you receive a message indicating that the segment is offline or invalid.

Screens for Segment/DASD Statistics

Four screens are used to display segment and DASD statistics:

- SEGMENT STATISTICS screen
- DASD STATISTICS screen
- DASD STATISTICS BY PAGE TYPE screen
- CHANGE SEGMENT STATUS screen

Each of these screens is described below.

Segment Statistics Screen

Function The SEGMENT STATISTICS screen displays all segments known to the Data Object Broker as defined in the DBGEN.

Example of Segment Statistics Screen

```
S6BADMB1   S6H1DOBB              SEGMENT STATISTICS              2006NOV29 10:13:15

 SEG NAME      MOD PAGES   FREE    % JRN SYS DEL D/S  WARN    READ   WRITE    HOLD THR
   0 VTH.S000 R/W  360K   224K   62 YES YES LOG   4 80/05    5514     142       0   7
   2 VTH.S002 R/W  216K   157K   73 YES NO  LOG   2 80/05      30      10       0  24
   3 VTH.S003 R/W   54K    53K   98 YES NO  LOG   1 80/05       0       0       0  24
   9 VTH.S009 R/W   18K  1428     7 YES YES LOG   1 80/05       6      18       0  24

     ENTER-REFRESH  PF2-DASD STATISTICS  PF7-BACK  PF8-FORWARD  PF11-MODIFY
```

Information Displayed

Column	Function
SEG	The segment number.
NAME	Symbolic name representing the segment name.
MOD	The current mode of operation: R/W: available for reading and writing R/O: available for reading only OFF: currently not available (offline)
PAGES	The total number of pages in all data sets within the segment. The page count appears in limited length output format, that is, if the number of pages exceed 5 digits, the value is factored by 1000 (KB), 1000000 (MB) or 1000000000 (GB).
FREE	The number of pages that are currently free. This value is in limited output 5-digit format.

Column	Function
%	Percentage of free pages available.
JRN	Whether (YES) journaling is active.
SYS	Whether (YES) the segment is a system-required segment, that is, it cannot be brought offline while the Data Object Broker is active.
DEL	The delete processing used, either physical (PHS) meaning the free pages are actually written to the data store or Logical (LOG) indicating that only the bitmap reflects a free page.
D/S	The number of data sets within the segment.
WARN	The first value is the threshold percentage at which warnings are produced to indicate a lack of free space within a segment. The second value is the rate at which the messages are produced. For example, if the display is 80/05 the first warning is produced when the segment is 80% full, addition messages are produced when the segment is 85%, 90% and 95% full.
READ	The current number of physical reads.
WRITE	The current number of physical writes.
HOLD	The current number of pages on hold.

Key Commands

Enter	Refresh data.
PF1	Display help information about the current screen.
PF2	Enter the DASD Statistics screen. Place the cursor on the desired segment number and press PF2.
PF3	Return to the main Administration menu.
PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.

PF11	Display the CHANGE SEGMENT STATUS screen. Place the cursor on the desired segment number and press PF11. **Note** The base segment (segment 0) cannot be modified online.
PF12	Exit from the Administration menu

DASD Statistics Screen

Function

The DASD Statistics screen presents statistical data for each page data set within a given segment.

Displaying the DASD Statistics Screen

This screen can appear for any active segment by placing the cursor beside the desired (online) segment on the SEGMENT STATISTICS screen and pressing PF2. If the segment number refers to a segment that is currently offline, the screen reappears with a message indicating the segment is offline.

Illustration of the DASD Statistics Screen

```
S6BADMB2  S6H1DOBB DASD STATISTICS FOR SEGMENT 000:VTH.S000 2006NOV29 10:16:39

          FIRST DSN HLQ.VTH.S000.PAGE1


 DATASET #            1            2            3            4
 VOLUME           OSBD62       OSBD20       OSBD53       OSBD48
 PAGES             90000        90000        90000        90000
 USED PAGES        34014        34430        34096        33189
 FREE PAGES        55986        55570        55904        56811
 READ               1450         1544         1391         1129
 WRITTEN              37           39           30           36
 MAX CONC              1            1            1            1
 ERRORS                0            0            0            0
 DELAY TIME            0            0            0            0
 DELAY CNT             0            0            0            0
 DELAY AVG.           12           14           12           13
 REQUESTED
 ACTIVE
 PENDING


    ENTER-REFRESH  PF2-BY PAGE TYPE  PF4-SEG STATUS  PF7-BACK  PF8-FORWARD
  NO MORE DATA TO BE DISPLAYED
```

Information Displayed

Row	Function
DATASET	The relative page data set numbers.
VOLUME	The symbolic names for the DASD device where the data set resides.
PAGES	The total page capacity of the data set. If the number of free pages is less than 15 percent of the total, the free value is highlighted.
USED PAGES	Number of pages used in the data set.
FREE PAGES	Number of pages free in the data set.
READ	The current number of physical reads.
WRITTEN	The current number of physical writes.
MAXCONC	The maximum number of outstanding concurrent I/O requests.
ERRORS	A count of the number of detected VSAM errors that can be retried.

Key Commands

Enter	Refresh data.
PF1	Display help information about the current screen.
PF2	Display the DASD statistics by page type.
PF3	Return to the main menu.
PF4	Redisplay the Segment Statistics screen with refreshed data.
PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF12	Exit from the Administration menu.

DASD STATS BY PAGE TYPE screens use the same data collection area. Unless a redisplay is requested on either screen, the data in the screen display is not refreshed.

DASD Statistics by Page Type Screen

Function

In the Pagestore, there is a variety of page types, each with a specific purpose. The DASD STATS BY PAGE TYPE screen divides the page types into three categories: DATA, INDEX, and OTHER, and displays physical reads and writes for pages containing TDS data within these categories.

Illustration of the DASD Statistics by Page Type Screen

```
S6BADMB3  S6H1DOBB DASD STATS BY PAGE TYPE SEG 000:VTH.S000 2006NOV29 10:16:39

              D A T A              I N D E X              O T H E R
 DS#        READ     WRITTEN        READ     WRITTEN        READ     WRITTEN
   1         988          12         107          10         355          15
   2        1088          16         103          13         353          10
   3         939          12         106           5         346          13
   4         845          22          70           4         214          10

    ENTER-REFRESH  PF2-DASD STATS  PF4-SEGMENT STATUS  PF7-BACK  PF8-FORWARD
```

Information Displayed

This screen contains the following information:

DATA	These pages contain the actual data rows.
INDEX	These pages contain index structures used to navigate indexes and data pages below them. There are a number of index types, including primary data index, secondary index, and parameterized table index.
OTHER	These pages are a catchall for pages not used for these purposes. They are primarily internal control pages.

Key Commands

Enter	Redisplay DASD STATS BY PAGE TYPE screen with refreshed data.
PF1	Display help information about the current screen.
PF2	Display the DASD STATISTICS screen.
PF3	Return to the main menu.
PF4	Redisplay the SEGMENT STATISTICS screen with refreshed data.
PF6	Print the current screen.
PF7	Scroll screen backwards.
PF8	Scroll screen forwards.
PF12	Exit from the Administration menu.

DASD STATISTICS and DASD STATS BY PAGE TYPE screens use the same data. Unless a redisplay is requested on either screen, the data is not refreshed.

Change Segment Status Screen

Function

The CHANGE SEGMENT STATUS screen provides a facility for authorized people to control the TIBCO Object Service Broker segments without using a batch job or the operator console.

Displaying the CHANGE SEGMENT STATUS Screen

If you have authority to control segments, place the cursor on the desired segment number and press PF11 to display the CHANGE SEGMENT STATUS screen.

Segments specified as system segments (for example, MetaStor segment 0) cannot be taken offline.

Illustration of the Change Segment Status Screen

```
S6BADMB4   S6H1DOBB            CHANGE SEGMENT STATUS          2006NOV29 11:43:39

                        SEGMENT 002:VTH.S002    TYPE=PAGESTORE

                                        CURRENT   REQUESTED
                      JOURNALLING       YES

                      MODE              READWRITE

                      STATUS            ONLINE


                         PF7  CHANGE JOURNALLING
                         PF8  CHANGE SEGMENT MODE
                         PF9  CHANGE SEGMENT STATUS


   NOTES 1-SEGMENT MUST BE OFFLINE TO EFFECT JOURNAL OR MODE CHANGES
         2-ONLY ONE ITEM CAN BE CHANGED WITHIN A SINGLE REQUEST


                  PF4-SEGMENT STATUS  PF11-ISSUE CHANGE REQUEST
```

Key Commands

PF1	Display help information about the current screen.
PF3	Return to the main menu.

PF4	Return to the SEGMENT STATISTICS screen without issuing a change request.
PF6	Print the current screen.
PF7	Toggle journaling between on and off: Y or N.
PF8	Toggle segment mode change request between READWRITE and READONLY.
PF9	Toggle segment status change request between ONLINE and OFFLINE.
PF11	Request the status change to be sent to the Data Object Broker. You are prompted for confirmation.
PF12	Exit from the Administration menu.

C. Lock Manager

Function Option C, LOCK MANAGER, displays a summary of the cumulative lock history.

Lock Manager Statistics Screen

```
S6BADMC1  S6H1DOBB          LOCK MANAGER STATISTICS          2006NOV29 11:52:20

      TRANSACTIONS                REQUESTS                 UPGRADE REQUESTS

 TOTAL          1138          TOTAL        37447          REQUESTS          924
 MAXIMUM          10          GRANTED      22118          THRESHOLD           0
 CURRENT           4          UPGRADE        924          ANCESTOR         2383
 BLOCKED           0          NO-OP        16022
                              FREE          1134

      RESOURCES                                             NO-OP REQUESTS

 TOTAL         18553                                       BLOCKED             0
 MAXIMUM         356                                       DEADLOCK            0
 CURRENT          61                                       IGNORED           169
                                                           REDUNDANT       15853



                  PF2-LIST HELD LOCKS BY TABLE      ENTER-REFRESH
```

Columns and Fields

The fields of the LOCK MANAGER screen are defined by column as follows:

Transactions

The most important statistic on the inquiry is TRANSACTIONS BLOCKED. This number informs you how many Execution Environment transactions were blocked due to record locking since TIBCO Object Service Broker last started.

TOTAL	A running count of all transactions.
MAXIMUM	The largest number of (simultaneous) transactions.
CURRENT	The current number of active transactions.
BLOCKED	Accumulated total of lock requests refused because of locks held by other transactions.

Requests

Requests are from the Execution Environments.

TOTAL	A running count of Lock Manager calls.
GRANTED	The total number of requests granted locks.
UPGRADE	The total number of requested locks upgraded from previously granted locks.
NO-OP	The requested lock is neither granted nor upgraded.
FREE	The number of resources held by transactions that were released.

Resources

Resources are logical locks.

TOTAL	A running count of all resources.
MAXIMUM	The largest number of resources held simultaneously.
CURRENT	The number of resources presently held.

Upgrade Requests

REQUESTS	The number of requests upgraded.
THRESHOLD	A running count of internal upgrades granted or blocked.
ANCESTOR	Upgrade of ancestor resources occurred due to an upgrade request.

NO-OP Requests

BLOCKED	The number of lock request conflicts with granted lock of another transaction; request rejected.
DEADLOCK	The total number of detected deadlocks.
IGNORED	The total number ignored lock requests
REDUNDANT	The total number of lock requests already granted at the requested level or stronger

List Locks on a Table

To list the locks held on a specific table, press PF2. This displays a screen prompting for the name of the table. Provide a table name and press ENTER to display the locks for the table, as shown below for the MESSAGES table. A message appears if no locks are found for the specified table.

```
S6BADMC2  S6H1DOBB  LOCKS HELD FOR TABLE MESSAGES          2006NOV29 11:52:20

   USER-ID         USER-ID         USER-ID         USER-ID         USER-ID
   DBA             USR00           ABC30

                  PF5-REFRESH     ENTER-TABLE NAME ____________________
  PLEASE SPECIFY TABLE NAME FOR HELD LOCKS INQUIRY
```

D. Buffer Pools

Function Option D, BUFFER POOLS, displays statistics about TIBCO Object Service Broker buffer pools.

Buffer Pools Statistics Screen

```
S6BADMD1  S6H1DOBB           BUFFER POOL STATISTICS          2006NOV29 11:55:32

 POOL                              IN-USE          QUEUED
 NAME      # GETS     # FREES    CURR    MAX    CURR    MAX    WAITS   ERRORS   BUFFERS
 BUFQ                                                                        120
 CLOG                                                                         25
 COMM         352         312      40     78                                851
 LOCK        1615        1610       5     16                                368
 NAMW                                                                         7
 PAGE      551699      551699            11                               7680
 SESS      128306      128306             2                                 34
 VRPL        6753        6753             7                               1038
 WORK       15101       15101             5                               1054
 XTAB         259         259             1                                 34

                        ENTER-REFRESH   PF7-BACK   PF8-FORWARD
```

TIBCO Object Service Broker Buffer Pools

These statistics concern the eleven different TIBCO Object Service Broker buffer pools, and are cumulative from when TIBCO Object Service Broker last started. The pools are:

BUFQ	Pool used to manage the hold page list for a session.
CLOG	Contingency log for commit coordinating processing.
COMM	Pool for active connections.
LOCK	Pool of lock working space.
NAMW	NAM work buffer.
PAGE	Pool of 4 KB work buffers.

SESS	Pool for active units of work, for example, queries or commits.
VRPL	VSAM RPL pool for Pagestore access.
WORK	Transient space used for certain system recovery actions.
XTAB	Pool for the external table definitions.

Column Headings

The column headings on the screen are described below:

POOL NAME	Symbolic name of the pool.
# GETS	The number of acquired buffers.
# FREES	The number of freed buffers.
IN-USE CURR	The number of buffers currently in use.
IN-USE MAX	The maximum number of buffers used in this pool.
QUEUED CURR	The number of requests waiting for a buffer.
QUEUED MAX	The maximum number of requests queued for this buffer.
WAITS	The number of waits for a buffer.
ERRORS	The number of errors associated with this buffer pool.
BUFFERS	The total number of buffers in this pool.

F. Profile Displays

Function — Option F, PROFILE DISPLAY SELECTION, displays a menu from which various profiles can be requested for display.

Profile Display Selection Screen

```
S6BADMF1  S6H1DOBB        PROFILE DISPLAY SELECTION         2006NOV29 11:56:48

                      1  MESSAGE LENGTH PROFILE

                      2  MESSAGE TURNAROUND TIME PROFILE

                      3  CONTINGENCY LOG I/O TIME PROFILE

                      4  REDOLOG RESPONSE TIME PROFILE

                      5  QUERY ECB WAIT PROFILE




                              ENTER ITEM NUMBER  _
```

Available Profiles

The following profiles are available and discussed below:

1. Message Length Profile
2. Message Turnaround Time Profile
3. Contingency Log I/O Time Profile
4. Redolog Response Time Profile
5. Query ECB Wait Profile

You can select any of the profiles by entering the associated number.

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF3	Return to the main menu.
PF4	Return to the Profile Display Selection screen.
PF6	Print the current screen.
PF12	Exit from the Administration menu.

1. Message Length Profile

Function The MESSAGE LENGTH PROFILE screen displays the numbers of messages received and sent in different size ranges.

Illustration of Message Length Profile Screen

```
S6BADMF3  S6H1DOBB MESSAGE LENGTH PROFILE                    2006NOV29 11:56:48
                                                               DELTA TIME
  LEN                  PERCENTAGE OF TOTAL                     COUNT      DELTA
 RECEIVES ----1----2----3----4----5----6----7----8----9----+
      64 |*****                                             |     13350
     128 |**************************                        |     72853
     256 |***************                                   |     41296
     512 |*                                                 |      2269
    1024 |*                                                 |        14
    2048 |*                                                 |         8
    4096 |*                                                 |        13
     >>> |                                                  |         0
    SENDS ----1----2----3----4----5----6----7----8----9----+
      64 |******************************                    |     80666
     128 |****                                              |     11553
     256 |*                                                 |      4348
     512 |***                                               |      7614
    1024 |****                                              |     10389
    2048 |*                                                 |      4298
    4096 |****                                              |     10873
     >>> |*                                                 |        49
                                                  RECEIVES      129803
                  ENTER-REFRESH  PF4-PROFILE SEL    SENDS      129790
```

LEN is the length of the message in bytes.

2. Message Turnaround Time Profile

Function MESSAGE TURNAROUND TIME PROFILE displays the response time in microseconds to query and commit messages received, grouped according to response time. The commit count displayed could include messages that do not cause updates to the display.

Illustration of Message Turnaround Time Profile

```
S6BADMF3  S6DCDOBA MESSAGE TURNAROUND TIME PROFILE          2007MAR30 08:08:45
                                                              DELTA TIME
 TIME                  PERCENTAGE OF TOTAL                    COUNT      DELTA
  QUERIES ----1----2----3----4----5----6----7----8----9----+
     250 |****************************************        |    116540
     500 |*****                                          |     14269
    1000 |*                                              |      1975
    2000 |*                                              |       904
    4000 |*                                              |       414
    8000 |*                                              |       203
   16000 |*                                              |       205
     >>> |*                                              |        93
  COMMITS ----1----2----3----4----5----6----7----8----9----+
     250 |*                                              |        50
     500 |*                                              |         4
    1000 |*                                              |         2
    2000 |***********                                    |      1739
    4000 |**********************************             |      5441
    8000 |*                                              |       243
   16000 |*                                              |        46
     >>> |*                                              |        31
                                                 QUERIES      134603
                    ENTER-REFRESH  PF4-PROFILE SEL COMMITS        7556

--------------------------------------------------------------------------------
?,              1 Sess-1    10.101.5.104                              1/1
```

3. Contingency Log I/O Time Profile

Function CONTINGENCY LOG I/O TIME PROFILE displays time in milliseconds for I/O taken by the contingency log, grouped according to I/O response time.

Illustration of the Contingency Log I/O Time Profile Screen

```
S6BADMF3  S6H1DOBB CONTINGENCY LOG I/O TIME PROFILE         2006NOV29 11:58:58
                                                              DELTA TIME
 TIME                  PERCENTAGE OF TOTAL                    COUNT       DELTA
   WRITES ----1----2----3----4----5----6----7----8----9----+
       2 |                                                 |          0
       4 |                                                 |          0
       8 |                                                 |          0
      16 |                                                 |          0
      32 |                                                 |          0
      64 |                                                 |          0
     128 |                                                 |          0
     >>> |                                                 |          0
          ----1----2----3----4----5----6----7----8----9----+




                                                  WRITES           0
                  ENTER-REFRESH  PF4-PROFILE SEL
```

4. Redolog Response Time Profile

Function REDOLOG RESPONSE TIME PROFILE displays time in milliseconds for I/O taken by the redolog.

Illustration of the Redolog Response Time Profile Screen

```
S6BADMF3  S6H1DOBB REDOLOG RESPONSE TIME PROFILE            2006NOV29 11:59:57
                                                               DELTA TIME
 TIME                   PERCENTAGE OF TOTAL                    COUNT      DELTA
   WRITES ----1----2----3----4----5----6----7----8----9----+
        1 |*********************************************  |        612
        2 |*                                              |         12
        4 |*                                              |          2
        8 |*                                              |          4
       16 |*                                              |          7
       32 |*                                              |          1
       64 |*                                              |          2
      >>> |                                               |          0
           ----1----2----3----4----5----6----7----8----9----+

                                                      WRITES         640
                  ENTER-REFRESH  PF4-PROFILE SEL
```

5. Query ECB Wait Profile

Function

Depending on the number of query tasks running in the Data Object Broker, as indicated by the value in the MAXQUERY Data Object Broker parameter, QUERY ECB WAIT PROFILE displays either of the following:

- If there is only one query task, a horizontal bar graph (on screen S6BADMF3) displays the depth of the queue (the number of messages waiting on the queue) versus the percentage of the total number of messages processed on the system.
- If there are multiple query tasks running, a vertical bar graph (on screen S6BADMF4) displays the depth of the queue versus the percentage of the total number of messages. On this screen, the information appears for each task and is identified as different sub-columns (1, 2, 3, and so on) under each depth column. You can press PF2 to toggle the values between percentages of the total and percentages of the delta change since the Enter key was last pressed.

A depth of more than 1 means that more than 1 message is waiting in the queue for that task and indicates that queries are being delayed due to resource constraints.

To run multiple query tasks, you need the TIBCO Object Service Broker Accelerator Pack for z/OS. For more information, refer to the MAXQUERY Data Object Broker parameter in *TIBCO Object Service Broker Parameters*.

Illustration of the Query ECB Wait Profile Screen – Single Query Task

```
S6BADMF3  S6H1DOBB QUERY ECB WAIT PROFILE                    2006NOV29 12:00:39
                                                               DELTA TIME
 CNT                  PERCENTAGE OF TOTAL                      COUNT      DELTA
    DEPTH ----1----2----3----4----5----6----7----8----9----+
       1 |************************                          |    139344
       2 |************************                          |    135493
       4 |                                                  |         0
       8 |                                                  |         0
      16 |                                                  |         0
      32 |                                                  |         0
      64 |                                                  |         0
     >>> |                                                  |         0
          ----1----2----3----4----5----6----7----8----9----+

                                                      DEPTH      274837
                    ENTER-REFRESH  PF4-PROFILE SEL
```

The system where this illustration was produced is a six-way system with 100 concurrent users.

Illustration of the Query ECB Wait Profile Screen – Multiple Query Tasks

```
S6BADMF4  S6PMDOBI QUERY ECB WAIT PROFILE (TOTAL)          2006NOV29 12:17:22
 DEPTH   1        2        4        8       16       32       64       64+
  100 12
   90 12
   80 12
   70 12
   60 12
   50 12
   40 12
   30 12
   20 12
   10 12
    0 123      123      123      123      123      123      123      123

   QUERY TASK 1  COUNT              471 DELTA              471 DELTA TIME
              2                      1                      1
              3                      0                      0



                    ENTER-REFRESH  PF2-DELTA TOGGLE  PF4-PROFILE SELECT
```

G. Updated Pages Per Transaction

Function

Option G, UPDATED PAGES PER TRANSACTION, displays the approximate number of pages updated by commit transactions. The bar graph shows the percentage of total transactions that fall within the specified range of updated pages, in 2% increments.

Updated Pages Per Transaction Screen

```
S6BADMG1   S6H1DOBB         UPDATED PAGES PER TRANSACTION      2006NOV29 12:25:51

   PAGES                PERCENTAGE OF TOTAL TRANSACTIONS
 UPDATED     ----1----2----3----4----5----6----7----8----9----+     TRANSACTIONS
     1      |*                                                |              19
     2      |************************************             |             680
     3      |********                                         |             155
     4      |*                                                |              23
     5      |**                                               |              33
     6      |*                                                |               3
     7      |*                                                |               7
     8      |                                                 |               0
     9      |*                                                |               2
    10      |                                                 |               0
    11      |                                                 |               0
    12      |                                                 |               0
    13      |                                                 |               0
    14      |                                                 |               0
    15      |                                                 |               0
   >15      |                                                 |               0

                                                        TOTAL             922
                                  ENTER-REFRESH
```

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF12	Exit from the Administration menu.

H. Physical CPU and I/O Statistics

Function Option H, PHYSICAL CPU AND I/O STATISTICS, displays the CPU and I/O statistics for the current life of a Data Object Broker and the Operator Session.

Physical CPU and I/O Statistics Screen

```
S6BADMH1  S6H1DOBB      PHYSICAL CPU AND I/O STATISTICS     2006NOV29 09:27:51

    ------- SYSTEM TOTALS ------                -- ADMIN SESSION TOTALS ---
                HHH MM SS.MICRO                             HHH MM SS.MICRO
    TCB TIME    000 00 27.927455                TCB TIME    000 00 00.064672
    SRB TIME    000 00 03.406916                SRB TIME    000 00 00.001406
    --------------------------                  --------------------------
    CPU TOTAL   000 00 31.334371                CPU TOTAL   000 00 00.066078
    I/O COUNT             12 813                I/O COUNT                101

    ------- SYSTEM DELTAS -------               - ADMIN TRANSMIT/RECEIVE --
                HHH MM SS.MICRO                             HHH MM SS.MICRO
    TCB TIME                                    TCB TIME    000 00 00.000208
    SRB TIME                                    SRB TIME    000 00 00.000063
    --------------------------                  --------------------------
    CPU TOTAL                                   CPU TOTAL   000 00 00.000271
    I/O COUNT                                   REAL TIME   000 00 00.000381
    ELAPSE TIME

                                ENTER-REFRESH
```

The System Deltas section measures the usage between the times you press Enter. The other sections are updated each time you press Enter.

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF12	Exit from the Administration menu.

I. User Activity

Function Option I, USER ACTIVITY, displays active user connections to the Data Object Broker on the ACTIVITY SUMMARY screen.

Activity Summary Screen

```
S6BADMI6  S6H1DOBB              ACTIVITY SUMMARY              2006NOV29 12:28:00

   USERID    VIA      QUERY    COMMIT       GET       PUT      READ     Q-CPU      IDLE
   ABC30     TSO      17118        82     78862       255       280      3559     11480
   USR00     TSO      15971        26     70543        63         9      2640        67
   DBA       TSO      13533       197     36406       244       211      2300        68
   DZY00     TSO       6241        19     39217        45      1168      3849      5934
   U00E6000 NDR           5         0        16         0        29         2    420583
   U00E6001 NDR           5         0         4         0         0         0    420583
   U00E6002 NDR           0         0         0         0         0         0    420583
   U00E6003 NDR           0         0         0         0         0         0    420583
   U00E6004 NDR           0         0         0         0         0         0    420582
   U00E6005 NDR           0         0         0         0         0         0    420582
   U00E6006 NDR           0         0         0         0         0         0    420582
   U00E6007 NDR           0         0         0         0         0         0    420581
   U00E6008 NDR           0         0         0         0         0         0    420581
   U00E6009 NDR           0         0         0         0         0         0    420581
   $00E6000 NDR           0         0         0         0         0         0    420575
   $00E6001 NDR           0         0         0         0         0         0    420575
   $00E6002 NDR           0         0         0         0         0         0    420574
   $00E6003 NDR           0         0         0         0         0         0    420574

 ENTER-REFRESH  PF2-ACTIVITY DETAIL  PF4-USER LIST  PF5-REGION LIST  PF9-RESORT
```

Headings

USERID	user IDs connected.
VIA	Connection mode.
QUERY	Number of queries.
COMMIT	Number of commits.
GET	Logical reads.
PUT	Logical writes.
READ	Physical reads.

Q-CPU	Estimated CPU times (seconds).
IDLE	Time since last activity (seconds).

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF2	Display the ACTIVITY DETAIL screen for the selected user ID.
PF3	Return to the main menu.
PF4	Display the USER LIST screen.
PF5	Display the REGION SELECTION LIST screen.
PF6	Print the current screen.
PF9	Sort entries in the ACTIVITY SUMMARY screen.
PF12	Exit from the Administration menu.

Available Screens

The following nested screens are available through the USER ACTIVITY screen:

- Activity Detail Screen
- Logical Locks Screen
- Region Selection List Screen
- Connections in Region Peer Screen

These screens are described below.

Activity Detail Screen

Function — Displays a detailed listing for a specified user.

To Display — Place the cursor on the desired ID in the USER ACTIVITY screen and press PF2. To sort by column, place the cursor on the column heading and press PF9.

Illustration of the Activity Detail Screen

```
S6BADMI2  S6H1DOBB        ACTIVITY DETAIL FOR ABC30           2006NOV29 12:40:44

  TYPE TSO USER         IDLE TIME 94:35    EST. CPU 00:00:03.6  ACCESS       XMS
  TERMINAL TCP04015     REGION TSO         JOB NAME ABC30       REF#         316
  START TIME    8:21:06 DATE   2006NOV29   TRACE    N

  CURRENT MESSAGE IDLE-NO MESSAGE               EXCEPTION
  SEGMENT        PAGE            TABLE

  QUERIES     17118                          GETS        78862
  COMMITS        82                          PUTS          255
  SERVERS         0                          READS         280
                                             RULES        2593

 STR  TRX ID    SERVERS
   1  00016550
   2  0001BC06



   ENTER-REFRESH PF2-LOCKS PF4-USER PF5-REGION PF9-SUMMARY PF11-OPER SERVICES
```

Sections on the Activity Detail Screen

The screen is divided into four sections:

- Session Identification (type, terminal, and communication)
- Current Message, activity snapshot
- Accumulated Statistics for queries, commits, and so on
- External Resource allocation by stream and type

The sections of the screen are described below:

Session Identification

TYPE	Identifies the type of region where the session connected, for example, TSO, CICS, Native Execution Environment.
IDLE TIME	The duration in minutes and seconds since the last action was requested.
EST. CPU	An estimate of CPU time accumulated since the connection was initiated. This value does not include some of the shared overhead costs and should not be used for accounting purposes.
ACCESS	Indicates the communication service being used: XMS — Cross Memory Services NET — VTAM communications TCP — TCP/IP communications This is determined by the TIBCO Object Service Broker Communication System. Refer to Appendix B, Configurations for Communications, on page 409 for more information.
TERMINAL	Identifies the terminal used to connect to TIBCO Object Service Broker if the ID is supplied at connection time.
REGION	Identifies the REGION or group code to which the session is attributed.
JOB NAME	Identifies the job name of a batch connection if the job name is supplied at connection time.
STEP	Identifies the job step of a batch connection if the step name is provided at connection time.
TRACE	(Y/N) — GTF trace active for users.

Current Message

CURRENT MESSAGE	Identifies the type of request that is currently in progress, for example, query or commit.

EXCEPTION	If the request is waiting on a TIBCO Object Service Broker resource, an exception is identified to indicate which resource.
SEGMENT, PAGE, and TABLE	If a request is in progress, these three values identify the current segment and page numbers and the table name last referenced by the request.

Accumulated Statistics

QUERIES	The number of QUERIES since the connection was made.
COMMITS	The number of COMMITS since the connection was made.
SERVERS	The number of requests to SERVERS since the connection was made.
GETS	The number of GETS since the connection was made.
PUTS	The number of PUTS since the connection was made.
READS	The number of READS since the connection was made.
RULES	The number of RULES executed since the connection was made.

External Resource Allocations

STR	Stream number.
TRX ID	Logical lock identifier.
SERVERS	The type and server connection ID of any external resources.

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF2	Display the LOGICAL LOCKS screen.
PF3	Return to the main menu.

PF4	Return to the ACTIVE USERS screen.
PF5	Display the REGION SELECTION LIST screen.
PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF11	Displays the Operator Services screen for this user.
PF12	Exit from the Administration menu.

Logical Locks Screen

Function — Displays the logical locks held by a selected user.

To Display — From the ACTIVITY DETAIL screen, place cursor on the desired stream and press PF2.

Illustration of the Logical Locks Screen

```
S6BADMI3  S6H1DOBB       LOGICAL LOCKS HELD BY ABC30        2006NOV29 12:54:16

   STR  TRX ID   SERVERS
     1  00016550

  TABLE NAME          TABLE NAME          TABLE NAME          TABLE NAME
 $GRAMMARS            MENU_ITEMS
 @CHARACTERMAP        MENU_TEMP
 @COLOURS             MESSAGES
 @LIBRARIES           PFKEY_SPECS
 @MONTH_CODES         REMIND
 @MONTHLIST           SEMANTIC
 @OBJECTMSG           SESSION_MENUS
 @SCREENCOLOURS       SESSMGR_MSG
 @USERSOPTIONS        STANDARD_BODY
 @WEEKDAYS            STANDARD_HEAD
 ALTERNATIVES         STANDARD_HIST
 APPOINTMENTS
 CALNDR
 CMD_HIST_TEMP
 GRAMMARS

                  PF4-ACTIVITY DETAIL  PF7-BACK  PF8-FORWARD
```

Key Commands

Press PF4 to redisplay the ACTIVITY DETAIL screen.

Region Selection List Screen

Function — The REGION SELECTION LIST screen displays the Execution Environment types.

To Display
- To display from the ACTIVE USER LIST screen, press PF4.
- To display from the ACTIVITY DETAIL screen, press PF5.

Illustration of Region Selection Screen

```
S6BADMI5  S6H1DOBB           REGION SELECTION LIST          2006NOV29 12:55:26

   COMBINED        NATIVE                          BATCH           TSO
   OPERATOR




     ENTER-REFRESH  PF2-SELECT REGION  PF4-USER LIST  PF5-ACTIVITY SUMMARY
```

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF2	Place the cursor beside the desired type of Execution Environment and press PF2 to display a list of connections within that group.

PF3	Return to the main menu.
PF4	Place the cursor beside the desired type of Execution Environment and press PF2 to display the ACTIVE USER LIST screen for that Execution Environment.
PF5	Display the ACTIVITY SUMMARY screen.
PF12	Exit from the Administration menu.

Connections in Region Peer Screen

Function — Lists all connections with the requested region code.

To Display — Position your cursor beside NATIVE in the REGION SELECTION LIST screen and press PF2.

Illustration of Connections in Region Native Screen

```
S6BADMI1  S6H1DOBB        CONNECTIONS IN REGION NATIVE      2006NOV29 13:03:22

   N-$00E6000      N-$00E6001      N-$00E6002      N-$00E6003      N-$00E6004
   N-$00E6005      N-$00E6006      N-$00E6007      N-$00E6008      N-$00E6009
   O-@OP00E60

   ENTER-REFRESH  PF2-ACTIVITY DETAIL  PF4-REGION LIST  PF5-ACTIVITY SUMMARY
```

User Types

The list of connections is sorted by user ID within user type. User type is indicated by the following letters:

(blank)	TSO user
N	Native Execution Environment.
L	Logging out.
D	Server (external database or peer TIBCO Object Service Broker).
C	CICS.
T	Other user type.
B	Batch user.
P	Inbound peer TIBCO Object Service Broker.

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF2	Place the cursor in front of the desired user and press PF2 to display the activity detail for the user.
PF3	Return to the main menu.
PF4	Redisplay the REGION SELECTION LIST screen.
PF5	Display the ACTIVITY SUMMARY screen.
PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF12	Exit from the Administration menu.

Operator Services Screen

Function — The OPERATOR SERVICES screen provides the ability to make changes for the specified user.

To Display — To display from the ACTIVITY DETAIL screen, press PF11.

Illustration of Operator Services Screen

```
S6BADMI7  S6H1DOBB      OPERATOR SERVICES FOR $00E6000      2006NOV29 13:04:54

                  SERVICE     GLOBAL  CURRENT  DESIRED  TOGGLE
                  TRACE         NO      NO       ___      PF4

                  PAGE SWEEP
                  - LOG         NO      NO       ___      PF5
                  - SMF         NO      NO       ___      PF7
                  - DUMP        NO      NO       ___      PF8
                  - LIMIT          0       0     ______

                  CANCEL                 NO       ___      PF9

                  FORCE                           ___      PF10
  WARNING:
  FORCE IS INTENDED TO ABNORMALLY TERMINATE A USER IF FOR SOME REASON AN
  OUTSTANDING CANCEL WAS UNABLE TO COMPLETE.  EVERY EFFORT IS TAKEN TO ENSURE
  THE USER IS REMOVED FROM THE SYSTEM AS CLEANLY AS POSSIBLE HOWEVER, THE
  NATURE OF THE FORCE CANNOT GUARANTEE TRANSACTION INTEGRITY.

  FORCE MUST BE USED WITH EXTREME CAUTION AND ONLY AFTER ALL OTHER MEANS
  HAVE BEEN EXAUSTED
                  PF2-ACTIVITY DETAIL  PF11-ISSUE COMMAND
```

Key Commands

Enter	Refresh the screen.
PF1	Display help information about the current screen.
PF3	Return to the main menu.
PF4	Toggle user-level tracing.
PF5	Toggle user-level Page Sweep Action reporting for the console log.
PF7	Toggle user-level Page Sweep Action reporting for the System Monitoring facility.

PF8	Toggle user-level Page Sweep Action reporting via an SVC Dump.
PF9	Prepare to issue a CANCELUSER command to quiesce the current session connection.
PF10	Prepare to issue a FORCEUSER command to forcibly remove the current connection from the Data Object Broker. Note the on-screen warning.
PF12	Exit from the Administration menu.

J. Installation/Configuration

Function

Option J, INSTALLATION/CONFIGURATION, displays information about the operating environment and configuration of the Data Object Broker. It also includes the hardware configuration and TIBCO Object Service Broker software level under which the Data Object Broker is running. This information is extracted from system control blocks and Data Object Broker parameters.

Installation and Configuration Screen

```
S6BADMJ1  S6DCDOBA       INSTALLATION AND CONFIGURATION     2011JAN07 03:05:48

                                   ENVIRONMENT
 O/S   zOS 1.11  CPU ID        F1A3  MODEL        2098  GMT ADJ   FFFF94BB

                                JOB INFORMATION
 TASKNAME DC52ADBA  ASID 0218  NODENAME S6DCDOBA          VERSION 600E006
 START 2011JAN07 03:03:40  COMMUNICATIONS:RELAY Y  CROSS MEMORY Y  VTAM Y
 ESTAE SUPPORT Y  OPERATOR SECURITY N  SWAPPABLE N  SRBMODE Z

                                STORAGE LIMITS
 CTAB RES    1024  CTAB SIZE    31744  XTAB SIZE    16384  RES PAGES      512
 LOCK BUF    8192  RSC MAX        128  NODE MAX       128  SCH AVG.         1
 TRACE EVENT   64  TRACE CTABLE   256  TRACE COMMIT  1024  TRACE QUERY   1024
 TRACE COMM.  768

                              PROCESS MANAGEMENT
 PAGES/CHPT 32767  TRX/CHPT     13499  CHPT INTVL     120  CONCURR. VSAM   24
 MAX THREADS  100  MAX USERS     1000  LOCK TIMEOUT    60  TIOT DELAY      25
 MAX QUERY TCB  1  DUPLICATE USERID Y  DUP REMOTE USER  N  CICS DSPL MAX    0
 SPIN INTVL     0  SPIN MBR  @CRASP    SPIN RUN      JOB
 SPIN DSN S6B.DC.OSB60.CNTL

                           TRANSACTION MANAGEMENT
 WORKING SET  999  READ LIMIT       0  DATA PG LIMIT  156  SIX PG LIMIT 00000
 PAGE SWEEP     0  PG SWEEP LOG     N  PG SWEEP SMF     N  PG SWEEP DUMP    N
```

```
                                MONITORING
GTFID          AA  SMF RECORD      FA  SMF-13 INT.      15  SMF-22 INT.      30
SMF-23 INT.     0  SMF-24 INT.       0  SMF-25 INT.       0  SMF-26 INT.       5
SMF-27 INT.     0  SMF-28 INT.       0  SMF-47 INT.       5  SMF-48 INT.      30
MSG TRACING     N  SERVICE TRACING   N  NETVIEW SUPPORT   N

                               WTO REPORTING
INTERVAL     1440  REMOTE ID LOCAL      SYSLOG LEVEL      #  CONSOLE LEVEL     F
PRINT CLASS     K  DELETE/SUPPRESS   N  MSG SUFFIX     DCA
  DESC:LOG 0000  INFORMATION 0000  WARNING 0000  ERROR 0000  ALERT/ACTION 0000
  ROUTE:   0020               0020          0020         0020                0020

                            SNAP/SVC DUMP REPORTING
SNAP HIGH QUALIFIER S6B.DC              CLASS A  VOLUME HRN070  UNIT HRNPOOL
DUMP ON QUERY ABORT Y  DUMP ON TRANSACTION ABORT N  DUMP ON INDOUBT TRX Y
NO MORE DATA TO BE DISPLAYED
```

Sections and Fields

The sections and the fields on this screen contain the following information:

ENVIRONMENT

This section presents information about the computer environment where the Data Object Broker is running. The data is extracted from the system Communications Vector Table.

O/S	Indicates the type of operating system. Extracted from the operating system.
CPU ID	The identification number for the CPU where the Data Object Broker is executing. Extracted from the operating system.
MODEL	The model number of the CPU where TIBCO Object Service Broker is executing. Extracted from the operating system.
GMT ADJ	The time difference between local time and GMT. Extracted from the operating system.

JOB INFORMATION

This section presents values pertaining to the TIBCO Object Service Broker job extracted from the Task I/O Table and runtime parameters. When applicable, the parameter names appear in parentheses at the end of the field descriptions.

TASKNAME	The name the Data Object Broker is executing under. This is extracted from the operating system.
ASID	The Data Object Broker address space identifier. This is extracted from the operating system.
NODENAME	A symbolic name to uniquely identify a Data Object Broker. When specifying a location in a rule, you use the “nodename” to identify the target location. (NODENAME)
VERSION	The version number of the TIBCO Object Service Broker software release. This value is constant and is set at the Data Object Broker source assembly time.
START	The date and time the Data Object Broker started. This information is extracted from the operating system.
RELAY	(Y/N) Indicates whether relay communication is supported. This value is determined by the TIBCO Object Service Broker Communication System. For more information, see The Relay File on page 424.
CROSS MEMORY	(Y/N) Indicates whether Cross Memory communication is supported. This is determined by the TIBCO Object Service Broker Communication System. Refer to Configuring Cross Memory Services on page 422 for more information.
VTAM	(Y/N) Indicates whether VTAM communication is supported. It is determined by the TIBCO Object Service Broker Communication System. Refer to Configuring VTAM Communications on page 411 for more information.
ESTAE SUPPORT	Specifies whether to enable the Data Object Broker Extended Specific Task Abnormal Exit (ESTAE) routine for handling abnormal terminations. (ESTAE)

OPERATOR SECURITY	Specifies whether the System Authorization Facility (SAF) calls are made to the site's security system. These calls are made for authorization of access to sensitive operational utilities requested from other than the z/OS operator console (for example, S6BTLADM, S6BTLCMD and S6BSPJEX). (SECUREADMIN)
SWAPPABLE	Specifies whether the Data Object Broker is marked as swappable after initialization. (SWAPPABLE)
SRBMODE	Specifies the mode in which query requests will be executed: • N — TCB mode • Y — SRB mode • Z — SRB mode with zIIP enabled

STORAGE LIMITS

This section shows information about the parameters that affect the major storage blocks used by the Data Object Broker.

Additional information about the Data Object Broker parameters associated with the following fields is available in *TIBCO Object Service Broker Parameters*. The parameter names appear in parentheses at the end of the field descriptions.

CTAB RES	The maximum number of table definitions, in CTABLEs, that can simultaneously reside in Data Object Broker storage. (CTABRESIDENT)
CTAB SIZE	The maximum size, in kilobytes, for CTABLEs, the internal representations of TIBCO Object Service Broker table definitions. (CTABLESIZE)
XTAB SIZE	The size, in kilobytes, of the buffer pool for XTABLEs, the internal representations of TIBCO Object Service Broker report and screen definitions. (XTABLESIZE)
RES PAGES	The number of 4 KB pages for data and index buffers, in thousands. (RESIDENTPAGES)
LOCK BUF	The number of additional 4 KB lock buffers to be acquired beyond the base amount. (LOCKBUFFERS)

RSC MAX	The maximum number of TIBCO Object Service Broker resources. (MAXRESOURCES)
NODE MAX	The maximum number of peer nodes. (MAXNODES)
SCH AVG.	The average number of schedule entries for each TIBCO Object Service Broker resource. (RSCSCHEDULES)
TRACE EVENT	The amount of space, in kilobytes, used for task event tracing.
TRACE CTABLE	The amount of space, in kilobytes, used for CTABLE management tracing.
TRACE COMMIT	The amount of space, in kilobytes, used for Commit system service tracing.
TRACE QUERY	The amount of space, in kilobytes, used for Query system service tracing.
TRACE COMM	The amount of space, in kilobytes, used for communications tracing.

PROCESS MANAGEMENT

This section presents the parameter values that affect the general processing of the Data Object Broker. The parameter names appear in parentheses at the end of the field descriptions.

PAGES/CHPT	The threshold, in number of changed pages, that triggers a checkpoint. (CHPAGELIMIT)
TRX/CHPT	The number of commits, issued from Execution Environments connected to this Data Object Broker, that cause it to take a checkpoint. (CHTRANLIMIT)
CHPT INTVAL	The maximum time interval in minutes between checkpoints, before the Data Object Broker triggers another checkpoint. (CHPTINTERVAL)
CONCURRENT VSAM	The maximum number of concurrent I/O requests to all page data sets. (MAXVSAMC)
MAX THREADS	The number of requests that the Data Object Broker can service at the same time. (MAXTHREADS)

MAX USERS	The maximum number of user sessions that the Data Object Broker allows to be active at one time. (MAXUSERS)
LOCK TIMEOUT	The wait interval in seconds before a lock request is retried if the initial request cannot be satisfied. (LOCKTIMEOUT)
TIOT DELAY	Reserved for TIBCO Object Service Broker use.
MAX QUERY	The number of parallel query TCBs. (MAXQUERY)
DUPLICATE USERID	Specifies whether to allow duplicate online user IDs to create sessions concurrently. (DUPUSERID)
DUP REMOTE USER	Specifies whether to allow connections with duplicate user IDs when either the existing or the connecting use is for an API process. (REMOTEDUP)
CICS DSPL MAX	Reserved for TIBCO Object Service Broker use.
SPIN INTVL	The maximum time interval between journal spins. If a spin is not scheduled during an interval due to a full journal or an operator request, the Data Object Broker schedules a spin at the end of this interval. Set this parameter's value to the length of that interval, in minutes. (SPININTERVAL)
SPIN MBR	The prefix for the member names that contain JCL for journal spins. (SPINMEMBER)
SPIN RUN	Specifies whether the journal spin JCL is run as a started task or batch job. (SPINOPTION)
SPIN DSN	The data set name from which SPIN jobs are submitted. It must be an existing partitioned data set containing member names set by the SPINMEMBER parameter. (SPINDSNAME)

TRANSACTION MANAGEMENT

This section shows the values used in the control of transactions as they are processed through the Data Object Broker.

WORKING SET	The maximum number of Pagestore pages that can be updated in a single Data Object Broker commit. (WORKINGSET)
READ LIMIT	The maximum number of pages that can be read in by a single Data Object Broker request. Transaction message requests that exceed the specified number of pages are aborted. (TRXREADLIMIT)
DATA PG LIMIT	The maximum number of data pages that can be updated within a transaction synchronization point process. (DATAPAGELIMIT)
SIX PG LIMIT	The maximum number of data pages a table can have for a secondary index to be built on it online. (SIXDATAPAGLIM)
PAGE SWEEP	The threshold number of logical page reads for page sweep reporting. (PAGESWEEPLIMIT)
PG SWEEP LOG	The page sweep reporting option. (PAGESWEEPACTION — the Write to log option)
PG SWEEP SMF	The page sweep reporting option. (PAGESWEEPACTION — the Write SMF record option)
PG SWEEP DUMP	The page sweep reporting option. (PAGESWEEPACTION — the Abend the transaction option)

MONITORING

This section presents the values that assist in the generation and collection of data to diagnose and monitor the usage and performance of the Data Object Broker. The parameter names appear in parentheses at the end of the field descriptions.

GTFID	The hexadecimal identifier for GTF trace records produced by the Data Object Broker when tracing is in effect. (GTFID)
SMF RECORD	The numerical identifier that this Data Object Broker uses when generating SMF records. (SMFRECORD)
SMF-## INT	The interval time between SMF record generation for active user sessions. Refer to the SMF# interval parameters described in *TIBCO Object Service Broker Parameters*.
MSG TRACING	Specifies whether to produce records for all query and commit requests received by the Data Object Broker. Use this parameter only in consultation with your TIBCO Support representative, as it can affect system performance. (MSGTRACE)
SERVICE TRACING	Specifies whether GTF records are to be created each time a TIBCO Object Service Broker system service event occurs. Use this parameter only in consultation with your TIBCO Support representative, as it can affect system performance. (SSTRACE)
NETVIEW SUPPORT	(Y/N) Indicates whether Netview is active in TIBCO Object Service Broker. Refer to TIBCO Object Service Broker Netview Support on page 253.

WTO REPORTING

The Data Object Broker is designed to generate a variety of console and SYSLOG messages to indicate processing steps. The parameter values presented in this section are used to control where and how these messages are generated and recorded. The parameter names appear in parentheses at the end of the field descriptions.

INTERVAL	The time interval after which the WTO output is to copied to a new data set. This parameter is in effect only if WTOCLASS is specified. A value of zero (0) disables this feature. (WTOINTERVAL)
REMOTE ID	The output destination for this Data Object Broker's JES WTO SYSOUT log data set. (WTOREMOTE)
SYSLOG LEVEL	The minimum level of messages to be written to the z/OS JES WTO SYSOUT file. All messages are written to the Data Object Broker JES SYSOUT file. (WTOSYSLOG) Message levels in descending order of importance are: A — action E — error W — warning I — information L — log only
CONSOLE LEVEL	The minimum level of Data Object Broker WTO messages to be suppressed rather than written to SYSLOG, or displayed on the z/OS operator console. The following message levels that can be suppressed: (WTOCONSOLE) A — action E — error W — warning I — information L — log
PRINT CLASS	The output class of a dynamically allocated JES SYSOUT file where the Data Object Broker copies WTO (write to operator) messages. (WTOCLASS)

DELETE/ SUPPRESS	The minimum level of Data Object Broker WTO messages to be suppressed rather than written to SYSLOG, or displayed on the z/OS operator console. The following message levels that can be suppressed: (WTODELETE) I — information L — log
MSG SUFFIX	The suffix to identify messages issued from a specific Data Object Broker. (WTOSUFFIX)
DESC.: LOG	The descriptor code to use for L (log) WTO messages. (DESCLOG)
INFORMATION	The descriptor code to use for I (information) WTO messages. (DESCINFORM)
WARNING	The descriptor code to use for W (warning) WTO messages. (DESCWARN)
ERROR	The descriptor code to use for E (error) WTO messages. (DESCERROR)
ALERT/ ACTION	The descriptor code to use for A (alert) WTO messages. (DESCACTION)
ROUTE: LOG	The route code for L (log) WTO messages. (ROUTLOG)[a]
INFORMATION	The route code for I (information) WTO messages. (ROUTINFORM)[a]
WARNING	The route code for W (warning) WTO messages. (ROUTWARN)[a]
ERROR	The route code for E (error) WTO messages. (ROUTERROR)[a]
ALERT/ ACTION	The route code for A (alert) WTO messages. (ROUTACTION)[a]

a. The value displayed on the Installation and Configuration Screen is a four-digits field. To interpret this field, convert each digit to binary. This gives a set of 16 bits, each of them, when on, from left to right, indicates that a route code is turned on. For example, a value of 9020 converts to 1001 0000 0010 0000 indicating that route codes 1, 4, and 11 are turned on.

SNAP/SVC DUMP REPORTING

This section presents the values that control the generation of SNAP and SVC dumps to aid in the diagnosis of transaction and process failures.

SNAP HIGH QUALIFIER	The high-level qualifiers for Data Object Broker snap dumps taken to a data set on DASD. (SNAPDSPREFIX)
CLASS	The JES output class for SYSOUT snap dumps generated by the Data Object Broker. (SNAPCLASS)
VOLUME	The volume serial of the DASD where the Data Object Broker can allocate snap dump data sets. (SNAPVOLSER)
UNIT	The generic name of a DASD pool to allocate DASD snap dump data sets. (SNAPUNIT)
DUMP ON QUERY ABORT	(Y/N) Specifies whether the Data Object Broker generates a transaction dump for failed query requests. (SNAPQUERY)
DUMP ON TRANSACTION ABORT	(Y/N) Specifies whether Data Object Broker transaction dumps are to be suppressed. (SNAPSUPPRESS)
DUMP ON INDOUBT TRX	(Y/N) Specifies whether a snap dump is to be generated when an in-doubt transaction is detected. (DBSNAP)

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for more information about generating SMF records.

Diagnostic Displays

P. Display Resident Page Buffer

Function

Option P, DISPLAY RESIDENT PAGE BUFFER, prompts you to input a relative resident page buffer number starting from 1 to the maximum defined for your Data Object Broker. It then displays the contents of this buffer.

The relative buffer number used for the input is displayed on the Resident Page Pool Directory screen, in the Buffer column. See V. Resident Page Pool Directory, page 356.

This display is intended for TIBCO Support use.

Display Resident Page Buffer Screen

```
S6BADMP1  S6ELDO RESIDENT PAGE DISPLAY                        2009JUL02 09:55:05

              BUFFER NUMBER         40

      0000    00000100  0E750100  18490000  0158C400   | .... .... .... ..D. |
      0010    A0900614  1302000E  A8000002  00010995   | .... .... y... ...n |
      0020    00550A7C  E4E2C5D9  6DC9D5C6  D603E2C3   | ...@ USER _INF O.SC |
      0030    D901D505  E2E3C1E3  E201D500  0006FAF0   | R.N. STAT S.N. ...0 |
      0040    F9F0F0F6  05C8E4D9  D6D50000  1EE28896   | 9006 .HUR ON.. .Sho |
      0050    A640A4A2  8599A240  93968787  85844096   | w us ers  logg ed o |
      0060    95A39640  A3888540  C4D6C200  010001D5   | nto  the  DOB. ...N |
      0070    00010001  0000760C  7CE4E2C5  D96DD3C1   | .... .... @USE R_LA |
      0080    E8D6E4E3  03E3C4E2  01D505E2  E3C1E3E2   | YOUT .TDS .N.S TATS |
      0090    01D50000  06FAF0F9  F0F0F605  C8E4D9D6   | .N.. ..09 006. HURO |
      00A0    D500003C  C39695A3  99969340  A3818293   | N... Cont rol  tabl |
      00B0    85408696  9940A388  8540C4C9  E2D7D3C1   | e fo r th e DI SPLA |
      00C0    E86DE4E2  C5D9E240  86A49583  A3899695   | Y_US ERS  func tion |
      00D0    4B404040  40404040  40404040  40404040   | .                   |
      00E0    00010001  D5010201  00010000  5A097CE4   | .... N... .... !.@U |
      00F0    E2C5D96D  D5E4D403  E2C3D901  D505E2E3   | SER_ NUM. SCR. N.ST |

                                  _____________________
ENTER-NEW BUFFER NUMBER PF5-REFRESH PF10-PREVIOUS PF11-NEXT
```

Key Commands

PF1	Display help information about the current screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF10	Previous buffer.
PF11	Next buffer.
PF12	Exit from the Administration menu.

S. Monitoring Process

Function

Option S, MONITORING PROCESS, displays current statistics and delta values for the current interval and accumulated values since the beginning of the process. During the setup process, you can choose a sampling interval of between 5 and 60 seconds.

Monitoring Setup Screen

The first screen presented for the Monitoring Process option is the MONITORING SETUP screen. Use it to set or modify the run characteristics of the monitor process. Set the desired interval, repeat count, print options, and display options.

```
S6BADMS1  OTSTSRV              MONITORING SETUP            2004JAN28 11:10:08


 1. SAMPLE INTERVAL           15   (5 TO 60 SECONDS)
 2. SAMPLE ITERATIONS        008   (2 TO 100 TIMES)
 3. PRINT INTERVAL SCREEN      Y   (Y-YES, N-NO)
 4. PRINT ACCUMULATIVE SCREEN  A   (A-ALL INTERVALS, E-AT END ONLY, N-NEVER)
 5. DISPLAY SCREEN             I   (A-ACCUMULATIVE, I-INTERVAL)




              ENTER-VALIDATE SETTINGS ONLY  PF2-START MONITORING
```

Starting Monitoring

Press Enter to validate any changes made to the setup. Pressing PF2 starts the monitoring and displays a screen similar to the one below.

Example of the Interval Monitor Screen

```
S6BADMS2   S6H1DO      INTERVAL MONITOR (INTERVAL 15 SECONDS) 2006NOV29 13:26:16

SND     210654     0  SPV         11     0  GET     809646     0  RD        7244     0
RCV     210667     0  COM       1171     0  PUT       2375     0  WRT        170     0
                      CHP         36     0  G4K      26479     0  RDO       1062     0
ON         379     0  FLE       8292     0  F4K        782     0  SMF       1440     0
OFF        338     0  APP     216374     0  DFC          1     0  JRL        170     0
                                             STL          0     0  EXC         14     0
MXU        150     0                        GFP        103     0  REX          0     0
MXC         78     0  PH0           0     0  PFP         76     0
TAM         13     0  PH1      207670     0  SYN       1041     0  SAD          0     0
                      PH2        1017     0                      WRP          0     0
PL          12     0  SRV          26     0  LCK      49995     0  DFP          1     0
GBF    1073008     0  OPR         452     0  ULK       3054     0
FBF    1073007     0                        RPL        183     0  RPL        276     0
                                             INS       1085     0  INS        960     0
SNP          0     0                        DEL        459     0  DEL        284     0
                                             CTB     211367     0  CTB        659     0
                                             RUL     155562     0  RUL       1362     0
SPT        220     0
CHT        190     0  RCV       0       0       0       0       0       0       0       0
CMT       2269     0  SND       0       0       0       0       0       0       0       0
FLT       1031     0  QRY       0       0       0       0       0       0       0       0
APT      37315     0  SYN       0       0       0       0       0       0       0       0
```

When the monitoring process has started, there is no intervention required. The screen is refreshed when the interval expires.

Information on the Monitoring Screen

The information on this screen consists of the same information as can be obtained with other selections from the GENERAL STATISTICS screen.

Data Fields

Refer to Fields on page 292 for information on the following data fields displayed in the Interval Monitor screen:

- MSG TRAFFIC
- DESTINATION
- LOGICAL
- PHYSICAL

Other Task counts

The bottom left part of the screen shows task counts for the following tasks:

SPT	Supervisor task.
CHT	Checkpoint task.
CMT	Communications task.
FLT	File task.
APT	Application task.

T. Data Object Broker Memory

Function

Option T, MEMORY DISPLAY, prompts you to supply a memory address and then displays the contents of that memory address in the MEMORY DISPLAY OF GLOBAL STORAGE screen.

This display is intended for TIBCO Support use.

Huron Memory Display of Global Storage Screen

```
S6BADMT1  S6H1DO MEMORY DISPLAY OF GLOBAL STORAGE          2006NOV29 13:29:56

              START=00016000  LEN=2000

      0000    C1D4C4E2  C4C3E3C8  E4D9D6D5  40404040   | AMDS DCTH URON      |
      0010    F2F0F0F6  F1F1F2F9  D5D6E503  0106333C   | 2006 1129 NOV. .... |
      0020    000D9F48  000FEFB0  00025FB0  00028F88   | .... .... .... ...h |
      0030    00026BF8  0003FFB0  00059F30  00025EE0   | .. 8 .... .... ..;\ |
      0040    00029FB0  00027FA8  0002AFB0  0003FF60   | .... .."y .... ...- |
      0050    1383E000  138C6000  0FC0B000  0FF1C000   | .c\. ..-. .{.. .1{. |
      0060    00000000  00000000  900FB010  58D0B260   | .... .... .... .}.- |
      0070    98ECD00C  07FE0000  00000000  00000000   | q.}. .... .... .... |
      0080    00000000  00000000  C8E4F0F0  800071C8   | .... .... HU00 ...H |
      0090    00014A88  007FF2C0  007E23A0  807E23A1   | ...h ."2{ .=.. .=.~ |
      00A0    00000000  00000000  40000000  0FC00000   | .... ....  ... .{.. |
      00B0    00000000  00023000  BFC77A72  DC13D240   | .... .... .G:. ..K  |
      00C0    00000006  C766D905  00017000  00016000   | .... G.R. .... ..-. |
      00D0    00000000  00015F88  00000000  0FC0A000   | .... ...h .... .{.. |
      00E0    00000000  00000000  BFC75B26  3FE56082   | .... .... .G$. .V-b |
      00F0    C3C8F1F0  8005C040  00061140  007DD828   | CH10 ..{  ...  .'Q. |

                                    ____________________
ENTER-NEW ADDR PF2-GLOBAL PF4-POOL LIST PF5-REFRESH
```

Key Commands

PF1	Display help information about the current screen.
PF2	Retrieve and display the TIBCO Object Service Broker global storage block. You can also display this by entering a **t** on the input line and pressing Enter.
PF3	Return to the main menu.
PF4	Display the POOL SELECTION LIST screen.

PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF10	Previous buffer.
PF11	Next buffer.
PF12	Exit from the Administration menu.

Pool Selection List Screen

Function — The Pool Selection List screen enables you to display the buffers of a selected pool one buffer at a time.

To Display — To view the POOL SELECTION LIST screen, press PF4 or enter a **p** on the input line and press Enter.

Illustration of the Pool Selection List Screen

```
S6BADMT2  S6H1DOBB          POOL SELECTION LIST             2006NOV29 13:29:56

                    POOL     DESCRIPTION

                    CLOG   CONTINGENCY LOG IMAGE BLOCKS
                    COMM   CONNECTED SESSIONS INFORMATION BLOCKS
                    LOCK   LOGICAL LOCK MANAGEMENT BLOCKS
                    PAGE   PAGE IMAGE PROCESSING BLOCKS
                    SESS   INFLIGHT TDS TRANSACTION BLOCKS
                    VRPL   VSAM RPL MANAGEMENT BLOCKS
                    WORK   TEMPORARY WORK SPACE BLOCKS
                    XTAB   NON-TDS TABLE DEFINITION IMAGE BLOCKS









                        ENTER-SELECT POOL ____    PF4-RETURN
```

Selections

The potential selections are described below:

CLOG	Contingency log image blocks.
COMM	Connected sessions information blocks.
LOCK	Logical lock management blocks.
PAGE	Page image processing blocks.
SESS	In-flight TDS transaction blocks.
VRPL	VSAM RPL management blocks.
WORK	Temporary work space blocks.
XTAB	Non-TDS table definition image blocks.

U. Page Image

Function Option U, PAGE IMAGE, enables you to view page images from an active Pagestore. The display consists of a page header, followed by a display in dump format.

This display is intended for TIBCO Support use.

Page Image Display Screen

```
S6BADMU1  S6H1DOBB          PAGE IMAGE DISPLAY            2006NOV29 14:14:51

 PAGE 0000-........ PREV ........ NEXT ........ ROWS .... SIZE .... TYPE ??????

      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |
      0000    00000000  00000000  00000000  00000000    | .... .... .... .... |

   PAGE/SEARCH TOKEN: ______________________________     (M FOR MAX SCROLL)
   PF2-READ  PF5-FIND  PF7-BACK  PF8-FORWARD  PF10-PREV PAGE  PF11-NEXT PAGE
  PLEASE SPECIFY A PAGE NUMBER; SEG-PAGE
```

Displaying a New Page

To display a new page, enter the segment s and page number (*p*) in the format *ssss-pppppppp* and press PF2. If the segment number is omitted, the same segment number as the last display is used.

Key Commands

PF1	Display help information about the current screen.
PF2	Read a new page. Enter the segment and page numbers, in hex notation, separated by a hyphen (-) in the format: *ssss-pppppppp*.
PF3	Return to the main menu.

PF5	Invoke the find function. Search tokens can be specified as follows:
	`x'...'`
	`c'...'`
	`'...'`
	`...`
	The search function supports first, last, previous, and next type attributes (only the first letter is required).

PF6	Print the current screen.
PF7	Scroll backwards.
PF8	Scroll forwards.
PF10	Move to the previous page.
PF11	Move to the next page.
PF12	Exit from the Administration menu.

V. Resident Page Pool Directory

Function

Option V, RESIDENT PAGE POOL DIRECTORY, displays the control table for the memory resident page images. The resident page pool directory is used by the Data Object Broker to manage the memory images of the TIBCO Object Service Broker pages. The user is prompted to supply the data set number of a segment.

This display is intended for TIBCO Support use.

Resident Page Pool Directory Screen

```
S6BADMV1   S6H1DOBB          RESIDENT PAGE POOL DIRECTORY       2006NOV29 14:17:15
                               SEGMENT 0000  DATASET 0000

 ENT    PAGE #     BUFFER   FLAG   XID   CHAIN      OWNER      QUEUE   QLEN LRU REFCNT
 001   00000000         44   00    00   00000000   00000000   13966000    0  00      0
 002   00000000         47   00    00   00000000   00000000   13966050    0  00      0
 003   00000000         55   00    00   00000000   00000000   139660A0    0  00      0
 004   00000000         63   00    00   00000000   00000000   139660F0    0  00      0
 005   00000000         66   00    00   00000000   00000000   13966140    0  00      0
 006   00000000         68   00    00   00000000   00000000   13966190    0  00      0
 007   00000000         74   00    00   00000000   00000000   139661E0    0  00      0
 008   00000000         76   00    00   00000000   00000000   13966230    0  00      0
 009   00000000         78   00    00   00000000   00000000   13966280    0  00      0
 00A   00000000         79   00    00   00000000   00000000   139662D0    0  00      0
 00B   00000000         83   00    00   00000000   00000000   13966320    0  00      0
 00C   00000000         84   00    00   00000000   00000000   13966370    0  00      0
 00D   00000000         91   00    00   00000000   00000000   139663C0    0  00      0
 00E   00000000         93   00    00   00000000   00000000   13966410    0  00      0
 00F   00000000         96   00    00   00000000   00000000   13966460    0  00      0
 010   00000000         97   00    00   00000000   00000000   139664B0    0  00      0
 018   00000015        101   00    00   00000000   00000000   00000000    0  00      0

ENTER NEW SEGMENT-DATA SET _______________   (M-MAX SCROLL)   PF7-BACK   PF8-FORWARD
```

W. Alert Messages

Function Option W, ALERT MESSAGES displays a full-screen listing of the latest type A messages. This listing is independent of job boundaries.

System Alert Summary Screen

```
S6BADMW1  S6H1DOBB            SYSTEM ALERT SUMMARY          2006NOV29 14:19:20

    DATE   TIME  PREFIX     MESSAGE TEXT
 2006NOV29 14:19 KX009 JRNL 002 SPIN:JOB=JOB04871 PG=       255 STATUS=00-00 DUR
 2006NOV29 14:19 KX008 JOURNAL 002 SPIN IN PROCESS
 2006NOV29 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV28 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV27 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV26 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV25 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV24 15:39 DB095 FREE SPACE SHORTAGE DETECTED FOR SEGMENT=VTH.S009 -  92%
 2006NOV24 08:44 KX009 JRNL 001 SPIN:JOB=JOB00682 PG=      6508 STATUS=00-00 DUR
 2006NOV24 08:44 KX008 JOURNAL 001 SPIN IN PROCESS
 2006NOV24 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  92% FULL
 2006NOV23 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV22 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV21 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV20 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV19 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV18 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL
 2006NOV17 00:00 KP058 WARNING  SEGMENT=VTH.S009 IS  91% FULL

                    ENTER-REFRESH  PF7-BACK  PF8-FORWARD
```

Z. Diagnostic Dumps

Function Option Z, DIAGNOSTIC DUMPS, enables you to create a variety of diagnostic memory dumps.

These dumps should be produced *only* on the advice of the TIBCO Support.

Diagnostic Dump Generation Screen

```
S6BADMZ1  S6H1DOBB        DIAGNOSTIC DUMP GENERATION        2006NOV29 14:20:08

                          1  COMMUNICATION AREA DUMP

                          2  FILE AREA DUMP

                          3  LOCK MANAGER DUMP

                          4  PAGE POOL AREA DUMP

                          5  DOB SVC DUMP GENERATION




                               ENTER ITEM NUMBER  _
```

Dump Options

From this menu you can create one of the following types of memory dumps:

COMMUNICATION AREA DUMP	Causes a dump of Communication buffers.
FILE AREA DUMP	Causes a dump of File memory.
LOCK MANAGER DUMP	Causes a dump of the memory used by the Lock Manager to be produced.
PAGE POOL AREA DUMP	Causes a dump of page pool memory.

Administration Controls

1. In-Doubt Transactions

Function — Option 1, IN-DOUBT TRANSACTIONS, displays a list of transactions involving service providers that could not be fully completed for some reason.

If the system is recycled before an in-doubt transaction is resolved, TIBCO Object Service Broker automatically reacquires the logical locks for the transaction during the restart process.

In-Doubt Transaction List Screen

```
S6BADM11  S6DCDOBA         IN-DOUBT TRANSACTIONS LIST         2008JUL09 06:13:38

                                      LOCAL                ORIGINATING
TRK    DATE    TIME    STAGE    STATUS  LOCK ID  USER ID  LOCK ID  NODE NAME
  1 2008JUL09 05:05 CONTNGT IN-DOUBT 0000023F CLARKD

                    PLACE CURSOR ON DESIRED TRACK AND HIT PF2 TO ZOOM
```

Screen Headings

TRK	Track Number — Identifies the contingency log entry.
DATE and TIME	The date and time when the commit was first saved on the contingency log.

STAGE	Stage or phase within the commit cycle that is currently being processed: PHASE 1 — Prepare-to-commit stage of a two-phase commit. PHASE 2 — Commit stage of a two-phase commit. CONTINGT (Contingent) — TDS updates are held pending confirmation of the service provider's update status.
STATUS	Identifies the current status of the contingency log PROGRESS — The commit cycle is continuing. M-COMMIT — The commit continues because of manual intervention. M-ABORT — The commit is aborted because of manual intervention. P-ABORT — The commit coordinator determined that the commit cycle should abort; however, either all resources have not been informed, or have not confirmed their aborts. HELD — The local TDS concerns of a commit are done; however, there are service provider commit or cleanup confirmations outstanding. CHECKING — The commit coordinator is currently attempting to resolve an in-doubt transaction. IN-DOUBT — A service provider is lost during the commit cycle; it is undetermined if the commit should continue.
LOCAL LOCK ID	A reference number generated by the Data Object Broker to relate concerns within an identifiable transaction.
USERID	An 8 character identifier used by TIBCO Object Service Broker security to determine access privileges.
ORIGINATING LOCK ID and NODE NAME	If the contingency log is created as a result of a commit coordinated by a different TIBCO Object Service Broker, the lock ID and node name displayed identify the transaction on the named TIBCO Object Service Broker node.

Other Available Screens

From the IN-DOUBT TRANSACTIONS LIST screen, you can display the following nested screens:

- The IN-DOUBT TRANSACTION DISPLAY screen
- The IN-DOUBT TDS DETAIL screen

These screens are described below.

In-Doubt Transaction Display Screen

Function — The IN-DOUBT TRANSACTION DISPLAY screen displays a detailed list of all resources involved in in-doubt transactions.

To Display — Place the cursor on the desired track in the IN-DOUBT TRANSACTIONS LIST screen and press PF2. A screen similar to the following appears:

```
S6BADM12  S6DCDOBA       IN-DOUBT TRANSACTIONS DISPLAY      2008JUL09 06:13:38

                                       LOCAL                ORIGINATING
 TRK    DATE    TIME   STAGE   STATUS  LOCK ID  USER ID  LOCK ID  NODE NAME
   1 2008JUL09 05:05 CONTNGT IN-DOUBT 0000023F CLARKD


                          TRANSACTION RESOURCE LIST

                   CONFIGURED   PROCESSING                      ACCESS    REMOTE
   TYPE  SERVER ID F.S. LEVEL   F.S. LEVEL    NODE               ID      TRACK

    TDS   LOCAL
    HRN   S6DCDOBB  FAIL SAFE 2  FAIL SAFE 1   S6DCDOBB         DAB00003



     PF2-TDS DETAIL PF4-IN-DOUBT LIST PF9-DELETE PF10-COMMIT PF11-ABORT
```

Key Commands

PF1	Display help information about the current screen.
PF2	Display the TDS DETAIL screen.
PF3	Return to the main menu.
PF4	Redisplay the IN-DOUBT LIST screen.
PF6	Print the current screen.
PF9	Manually delete the transaction from this system, if authorized.
PF10	Manually commit the transaction, if authorized.

PF11	Abort the transaction, if authorized.
PF12	Exit from the Administration menu.

In-Doubt TDS Detail Screen

Function — The IN-DOUBT TDS DETAIL screen displays TDS concerns for the contingency log Entry shown on the IN-DOUBT TRANSACTION DISPLAY screen.

To Display — Press PF2 from the IN-DOUBT TRANSACTION DISPLAY screen.

```
S6BADM13  S6DCDOBA           IN-DOUBT TDS DETAIL             2008JUL09 06:14:16
                                        LOCAL                ORIGINATING
 TRK     DATE    TIME    STAGE    STATUS  LOCK ID  USER ID  LOCK ID  NODE NAME
   1 2008JUL09 05:05 CONTNGT IN-DOUBT 0000023F CLARKD

                     TDS PENDING                                TDS PENDING
TABLE NAME         INST  DELT  REPL      TABLE NAME         INST  DELT  REPL
CLXWRITE                                 DJC_PEER             1     0     0

                          PF2-IN-DOUBT DISPLAY  PF4-IN-DOUBT LIST
```

2. Operator Functions

Function Option 2, OPERATOR FUNCTIONS, displays the most recent log messages.

Message Log Display Screen

```
S6BADM21  S6H1DOBB           MESSAGE LOG DISPLAY            2006NOV29 14:22:31

13:18:07 S6BKC017L-V50 DBA      LOGGED OFF #=000041 TYPE=TSO USER COMM=XMS     SO
13:18:07 S6BKC017L-V50 @OP008D0 LOGGED OFF #=000040 TYPE=OPERATOR COMM=XMS     SO
13:28:40 S6BKC017L-V50 USR00    LOGGED OFF #=000039 TYPE=TSO USER COMM=XMS     SO
13:28:40 S6BKC017L-V50 @OP00CE0 LOGGED OFF #=000038 TYPE=OPERATOR COMM=XMS     SO
14:11:00 S6BKC017L-V50 USERP    LOGGED OFF #=000033 TYPE=OPERATOR COMM=XMS     SO
14:14:51 S6BKC016L-V50 USERP    CONNECTED  #=000033 TYPE=OPERATOR COMM=XMS     SOU
14:17:08 S6BKC017L-V50 USR00    LOGGED OFF #=000036 TYPE=TSO USER COMM=XMS     SO
14:17:08 S6BKC017L-V50 @OP01090 LOGGED OFF #=000035 TYPE=OPERATOR COMM=XMS     SO
14:18:45 S6BKC017L-V50 USERP    LOGGED OFF #=000033 TYPE=OPERATOR COMM=XMS     SO
14:19:00 S6BKP005L-V50 MODIFY S6H1DOBB SPINS=I
14:19:00 S6BKP006I-V50 MODIFY COMMAND ACCEPTED
14:19:00 S6BKX051L-V50 START CHECKPOINT -      822 14:19:00 LOCAL
14:19:01 S6BKX052L-V50 END   CHECKPOINT -      822 14:19:00 LOCAL COUNT=      77
14:19:01 S6BKX008A-V50 JOURNAL 002 SPIN IN PROCESS
14:19:02 S6BKC016L-V50 OSBBSTC  CONNECTED #=000033 TYPE=OPERATOR COMM=XMS     SOU
14:19:02 S6BKX009A-V50 JRNL 002 SPIN:JOB=JOB04871 PG=       255 STATUS=00-00 DUR
14:19:02 S6BKC017L-V50 OSBBSTC  LOGGED OFF #=000033 TYPE=OPERATOR COMM=XMS     SO
14:19:02 S6BDB090I-V50 OPEN S6B.VTH.JRNL2
14:19:17 S6BKC016L-V50 USERP    CONNECTED #=000033 TYPE=OPERATOR COMM=XMS     SOU

                       ENTER-REFRESH    PF2-COMMAND LIST
```

Issuing Operator Commands

Press PF2 to display a list of operator commands for the Data Object Broker. To issue a command:

- Press Tab to place the cursor on the desired command and press Enter.
- Type a command on the input line and press PF10 to issue the command.

You are prompted to press PF11 to confirm the command.

Illustration of the Operator Commands Screen

```
S6BADM22  S6DCDOBA              OPERATOR COMMANDS              2011JAN07 03:06:58

    BWOSTATUS                    CANCELTERMID   (I)           CANCELUSER     (I)
    CHECKPOINT                   COMRESTART                   CRSTATUS
    CS              N/A          DBJRNLOFF      (B)           DBJRNLON       (B)
    DBOFFLINE       (B)          DBONLINE       (B)           DBREADONLY     (B)
    DBREADWRITE     (B)          DBSEGMENTSTAT  (B)           FORCEUSER      (I)
    FREEZE                       GTFSTATUS                    JOURNALOFF     N/A
    JOURNALON       N/A          JOURNALSTATUS                MEMBERLIST
    NETSTART                     NOTRACE        (I)           PARAMETER      N/A
    PEERCANCEL      N/A          QUIESCE                      RESUME
    SHUTDOWN                     SPINENABLE     N/A           SPINLOADJCL    N/A
    SPINSUBMIT=N                 SPINSUBMIT=I                 SPINSUBMIT=C
    STARTPEER       N/A          STOPSERVER     (E)           SWEEPACTION    N/A
    SWEEPLIMIT      N/A          SWITCH         N/A           TIMESTAMP
    TRACEID         (I)          UNFREEZE                     USERCOUNT      (I)
    USERLIST        (I)          USERMAX        N/A           WTOPRINT

 INPUT COMMAND:  ____________________________________________________________

           ENTER-ISSUE SELECTED COMMAND   PF10-ISSUE INPUT COMMAND
```

3. Resource Management

Function Option 3, RESOURCE MANAGEMENT, lists the TIBCO Object Service Broker resources you can manage with the Administration menu. To update TIBCO Object Service Broker resources through the Resource Management facility, you must have TIBCO Object Service Broker Administrator or Operator authority.

What is a TIBCO Object Service Broker Resource?

A TIBCO Object Service Broker *resource* is an interface to an environment managed by the Data Object Broker. For example:

- A connection to or from another Data Object Broker
- A connection to an external DBMS from the Data Object Broker
- A connection to a peer server or a Service Gateway for Files

Data Object Broker Parameters and Resource Management

The following Data Object Broker parameters affect the processing of resource management data:

DYNAMICRESOURCE	Specifies whether to allow resource entries to be created dynamically.
MAXNODES	Maximum number of peer nodes.
MAXRESOURCES	Maximum number of resources.
RSCSCHEDULES	Average number of schedule entries for each resource. Each line in a schedule entry counts as an entry when estimating a value for RSCSCHEDULES.

Refer to *TIBCO Object Service Broker Parameters* for more information about these parameters.

RESOURCE File

The data needed for resource management is stored in a standard VSAM Keyed Sequenced Data Set (KSDS) referred to as a *repository*. Its record size is 128. It can be created, backed up, and reorganized using IDCAMS when the Data Object Broker is offline. It must be defined in the DBDLIB.

Resource Types

Resources are identified by a 3-character code:

Code	Resource Type
Table type (for example, ADA, DAT, DB2)	External database server.
API	• Remote peer servers that provide rules access on a remote TIBCO Object Service Broker node. • Service Gateway for Files

Code	Resource Type
HIN	Inbound Data Object Broker communications, identifying connections from a remote TIBCO Object Service Broker node to a local TIBCO Object Service Broker node.
HRN	Outbound Data Object Broker communications, identifying connections from a local TIBCO Object Service Broker node to a remote TIBCO Object Service Broker node.

The RESOURCE TYPE LIST screen displays all the available resource types. The Resource Manager supports up to 32 different resource types.

Peer server and external database server resources are established by running batch jobs or started tasks. Use the Resource Manager to schedule when these resources are available. The Resource Manager does not start these tasks for you.

Inbound (HIN) and outbound (HRN) resources defined for peer TIBCO Object Service Broker communications must be defined at both ends of the connection.

Resource Groups

Connections within each resource type can be grouped into connections that have the same characteristics and are functionally equivalent to each other. Up to 512 connections can be defined within a single group.

The combined resource type and the resource group are referred to as the *resource name* used for scheduling resources or displaying their details. For example, an inbound connection (HIN resource type) in the resource group called `OSTARSRV1` would have the resource name of `HINOSTARSRV1`.

The RESOURCE GROUP LIST screen displays all the available resource groups.

Use resource groups to define common characteristics for connections. For example, you should define a group of resources for each server ID in an external database server. To enable connections between Data Object Brokers, define a resource group for each Data Object Broker.

Wildcard Processing

The value "????????" is a reserved resource group. If you specify this value, the Resource Manager generates a definition for a resource at connection time, using dynamic resource naming.

Resource Type List Screen

Function The RESOURCE TYPE LIST screen shows the available resource types.

Illustration of the Resource Type List Screen

```
S6BADM31  S6H1DOBB            RESOURCE TYPE LIST            2006NOV29 14:35:55

                  ADA           API           DAT           DB2

                  HIN           HRN           IDM           IMS




                                TYPE ___   GROUP _________

 ENTER-GROUPS PF2-DETAIL PF4-NODES PF5-ADD PF9-TURNAROUND PF10-SCHED PF11-STOP
```

Key Commands

Enter	Display the groups within a selected resource type. Select a resource type by entering its name or positioning the cursor on the name. In some emulators, you can also place the mouse pointer over the type name and use the right mouse button to select a type.
PF1	Display help information about the current screen.
PF2	Display details about the selected resource. Enter a valid TYPE and GROUP to display the RESOURCE DETAIL screen.
PF3	Return to the main Administration menu.
PF4	Display all TIBCO Object Service Broker node names known to the Resource Manager. The NODE NAME LIST screen appears.

PF5	Add a new entry in the resource detail list. The RESOURCE DETAIL screen appears in input mode. If no resources are defined when selecting RESOURCE MANAGEMENT from the main Administration menu, the RESOURCE DETAIL screen appears in input mode.
PF9	Display a summary profile of message turnaround times for various resource types. The MESSAGE TURNAROUND TIME screen appears.
PF10	Display a list of all defined schedule names known to the Resource Manager. The DEFINED SCHEDULE LIST screen appears.
PF11	Stop all connections of all paths within a selected resource type. You are prompted to confirm the request.
PF12	Exit from the Administration menu.

Managing TIBCO Object Service Broker Resources

From the RESOURCE TYPE LIST screen, you use the following screens to manage TIBCO Object Service Broker resources:

- Resource Group List Screen
- Resource Detail Screen
- Node Name List Screen
- Message Turnaround Time Screen
- Defined Schedule List Screen

The sections below describe these screens.

Resource Group List Screen

Function — The RESOURCE GROUP LIST screen shows the available resource groups within a selected type.

To Display

- From the RESOURCE TYPE LIST screen, select a resource type and press Enter.
- From the RESOURCE DETAIL SCREEN or the PATH LIST SCREEN, press PF4.

If more than 133 groups are defined within a type, 28 rows of data appear on a model 3 display, and 39 rows of data appear on a model 4 display.

Illustration of the Resource Group List Screen

```
S6BADM32  S6H1DOBB    RESOURCE GROUP LIST FOR TYPE = API    2006NOV29 14:38:08

   DEFAULT0    FILEGATE    GEORGETN    PICTON

 INPUT GROUP _________ ENTER-DETAIL PF4-TYPE LIST PF5-ADD GROUP PF11-STOP GROUP
 NO MORE DATA TO BE DISPLAYED
```

Key Commands

Enter	Display the resource detail for a selected resource group. Select a resource group by entering its name or positioning the cursor on the name. In some emulators, you can also place the mouse pointer over the group name and use the right mouse button to select a group.
PF1	Display help information about the current screen.
PF3	Return to the main Administration menu.
PF4	Return to the RESOURCE TYPE LIST.
PF5	Add a group. The RESOURCE DETAIL screen appears in input mode. The type is set to the resource type displayed in the RESOURCE GROUP LIST screen. If no resources are defined when selecting RESOURCE MANAGEMENT from the main Administration menu, the RESOURCE DETAIL screen appears in input mode.
PF11	Stop all connections within a selected resource group. You are prompted to confirm the stop request.
PF12	Exit from the Administration menu.

Resource Detail Screen

Function

The RESOURCE DETAIL screen displays the following resource information:

- Defined characteristics of the resource: for example, resource type and group, commit level, in-doubt transactions, node name
- Configuration of the online and common connection paths
- Statistics by access type: how many paths are in use and the number of transactions against them
- The resource schedule that applies to those paths

To Display

- From the NODE NAME LIST screen, place the cursor on the desired node name. Press Enter for outbound paths (HRN resource type) and press PF2 for inbound paths (HIN resource type).
- From the RESOURCE TYPE LIST, select a resource type and a resource group and press PF2.
- From the RESOURCE GROUP LIST, select a resource group and press Enter.
- From the PATH LIST SCREEN, press PF5.

- From the DEFINED SCHEDULE LIST, select a schedule and press PF2.
- From the RESOURCE SCHEDULE screen, or the USERS OF SCHEDULE screen, press Enter.

Illustration of the Resource Detail Screen

```
S6BADM33  S6H1DOBB     RESOURCE DETAIL FOR API DEFAULT0     2006NOV29 14:38:08

 INTERMEDIATE ROLLBK  Y EARLY RELEASE    Y LAST USER REUSE    Y COMMIT LEVEL  2
 RETRY INTERVAL       0 TP NAME            USER ID PREFIX       FAILURES      0
 NODE  AMDSDCTHURON     INDOUBTS          N MONITOR/SMF        N

                       CONNECTIONS      IN-USE         TRX          MESSAGE
                      CUR  MAX  LMT    CUR  MAX       COUNT          COUNT
            ONLINE      0    0    0     0    0          0              0
            COMMON     10   10   10     0    0          0              0

 SCHEDULE NAME COMMON

          APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN    DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y   Y                00:00    N          10




  ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
NO MORE DATA TO BE DISPLAYED
```

Use the RESOURCE DETAIL screen to monitor resource path connections.

The top section of the screen presents the defined characteristics of the resource. The middle section shows statistics by access type. The lower portion displays the current schedule of connection limits.

Resource Characteristics

The characteristics section of the screen displays the following fields:

INTERMEDIATE ROLLBK	This flag indicates whether this resource expects to receive a “sync” message following a rollback request to terminate a transaction. The setting of this value is specific to each resource type. Refer to the manual for the server to determine what it should be. Valid values are “Y” (yes), “N” (no), and “?” (wildcard). A wildcard indicates that the resource takes on the characteristic of the connection. We recommend that you not use the wildcard setting for this characteristic under normal conditions.
EARLY RELEASE	The early release feature enables resources that were not updated to be released early in the commit cycle. The early release shortens the duration of the commit cycle by reducing the number of contributors. It also releases the resource path so that it can be reassigned sooner. **Note** `EARLY RELEASE` should not be used with paths that are generated for specific batch jobs or when there is only one path available. Valid values are “Y” (yes), “N” (no), and “?” (wildcard).
LAST USER REUSE	This features gives selection preference to paths that were most recently used by the current user. Some resources can bypass part of the security processing if the same user processes a subsequent transaction. Valid values are “Y” (yes), “N” (no), and “?” (wildcard).
COMMIT LEVEL	The commit level indicator identifies the highest commit level a resource is capable of supporting. The three levels are “0” (serial), “1” (contingent or Fail Safe 1), and “2” (two phased or Fail Safe 2). DBMS resources should be set to “1” and peer Data Object Brokers and peer rules servers to “2”. **Note** To retain data integrity in a commit group with more than 1 update location, level “0” should not be used.

The following group of characteristics are used only for the resources for which the Data Object Broker generates connections.

RETRY INTERVAL	If the connection fails and there are no active connections to the target the retry interval specifies a wait time in minutes (1-60) until the connect attempt is retried. If “0”, there is no retry attempt.
TP NAME	When multiple Data Object Brokers are running on a non-z/OS platform a TP Name must be supplied so the communications subsystem can differentiate between them.
USER ID PREFIX	The generated unique user ID of the path. To avoid conflicts and enhance identification it is recommended that the user ID prefix be unique throughout the configuration.
FAILURES	The number of transactions that failed because there were no paths available for the resource. That is, you did not have enough resources to meet the demand. To diagnose a remote access failure, check for local failures under HRN and remote failures under API.
NODE	Specifies the logical node name (up to 16 bytes) of peer TIBCO Object Service Broker resources. The Node is required input for all “HRN” and “HIN” type resources. It must be unique throughout the configuration.
INDOUBT	Indicates whether there are in-doubt transactions against the displayed resource.
MONITOR/ SMF	If the SMFRECORD and SMF28INTERVAL Data Object Broker parameters are set, to capture interval server usage statistics, set the monitor/smf value to “Y”. By default, the statistics are not collected.

Monitoring Path Connections

The statistics displayed for the resource are identified as either *online* or *common*, where:

Online resources	Can be used only by transactions run from a terminal.

Common resources	Can be accessed by batch jobs or terminal-based transactions.

The following path usage statistics appear:

CONNECTIONS	The current, maximum concurrent, and limit connection counts.
IN-USE	The **MAX** column shows how many paths were used in the life of the current Data Object Broker. The **CUR** field shows the number of paths currently in use.
TRX COUNT	The number of transactions associated with the resource.
MESSAGE COUNT	The total number of messages sent to the resource.

You should review the **CONNECTIONS** and **IN USE** counters regularly to ensure the appropriate number of connections are scheduled to meet the demand.

Updating Resource Entries

Press PF11 on the RESOURCE DETAIL screen to update or delete the resource entry. You are prompted to confirm your changes. You can edit these fields in input mode:

Resource Type	Specifies the kind of connection on this resource to the remote component. Valid values are: HRN — For an outbound connection to a peer Data Object Broker HIN — For an inbound connection to a peer Data Object Broker API — For a connection from a peer server or a Service Gateway for Files DB2, IDM, and so on — For connections to external database servers. This field usually has the same value as the TIBCO Object Service Broker table type. An outbound connection is used if the local Data Object Broker issues data access requests to the remote peer. Inversely, an inbound connection is for data access requests coming from a remote peer. If the same remote peer requires both outbound and inbound connections, two resource detail entries are required.
Resource Group	The name of the resource group.
INTERMEDIATE ROLLBK	Indicates whether the resource expects to receive a sync message following a rollback request. Valid values are Y or N. You should not use the wildcard (?) character under normal conditions.

EARLY RELEASE	When a service provider is released at the end of a transaction, it is released either in series or in parallel. Early release is recommended. The early release option causes external resources that were not updated within the transaction to be released simultaneously, that is, in parallel. This decreases turnaround time and improves response time performance for interactive users because the effective commit group is reduced. If early release is selected, the slight delay in server release can potentially increase server-not-available failures if there are too few servers connected. It usually takes longer for server release and acknowledgment to be processed than for TIBCO Object Service Broker to turn around the user transaction end (that is, the user application could be ready for the next transaction before the server is again generally available for it).
LAST USER REUSE	With some external database management systems, it is more efficient if the same server is used to process consecutive transactions from the same user. With DB2, for example, the server permits bypassing of some processing related to security when a new transaction is initiated via a server whose immediate previous user was the same as the current one. Other servers, however, could be unaffected by such reuse. Enabling this facility for such servers yields no benefit, and, in fact, results in slightly reduced efficiency.
COMMIT LEVEL	This indicator specifies the highest commit protocol level that an external resource can support. Valid values are: 2 — Two-phase commit protocol. Recommended for use with other Data Object Brokers and with peer servers 1 — Contingent commit. Recommended for external database servers. 0 — Serial Commit. Not recommended for any connection. Use only if the external resource is unable to support Fail Safe levels 1 or 2.

RETRY INTERVAL	If a connection fails and there are no active connections to the target, RETRY INTERVAL specifies the wait time in minutes (1-60) before retrying a connection. A setting of 0 indicates no attempt to retry. Used only for the resources that the Data Object Broker generates connections for (resource type HRN).
TP NAME	Identifies a Data Object Broker on a non-z/OS platform running multiple Data Object Brokers.
USER ID PREFIX	When a connection request is generated to a remote resource, a user ID is generated to identify the connection. The user ID is prefixed by these three characters. When a connection to a resource is generated by the Data Object Broker, the user ID associated with that connection must be unique. The actual user ID contains a five-digit sequence number appended to the specified three-character prefix; for example, `T2P00001`.
NODE	The logical node name, up to 16 bytes, of a peer TIBCO Object Service Broker resource. A node name is required for outbound (HRN) and inbound (HIN) resources. TIBCO Object Service Broker rules use the node name to indicate a target Data Object Broker, for example `GET TABLE ('nodename');`. It must be unique throughout the configuration.
SCHEDULE NAME	Name of a resource schedule that controls connection limits. The name can be up to 11 characters. It can be different from the resource name, and if a schedule is specific to a resource, make the schedule name the same as the resource name. For example, an inbound connection (HIN resource type) in the resource group called `OSTARSRV1` would have the resource name of `HINOSTARSRV1`. To reduce maintenance effort, you should use generic schedule names (for example, `WEEKDAYS05`, `WEEKEND06`) and define shared schedules where possible.
DELETE	Appears when the RESOURCE DETAIL screen is in input mode. To delete the resource detail entry, set to Y. The DELETE field is not visible in browse mode.

Key Commands

Enter	Display the connected paths within a named resource.
PF1	Display help information about the current screen.
PF2	Return to the RESOURCE TYPE LIST screen.
PF3	Return to the main Administration menu.
PF4	Display the RESOURCE GROUP LIST screens.
PF5	Display the NODE NAME LIST screen.
PF9	Start a connection to another outbound peer if the current connection count for online or common accesses is below the maximum connection limit.
PF10	Display a list of all defined schedule names known to the Resource Manager. The DEFINED SCHEDULE LIST screen appears.
PF11	Toggle browse and input mode of the RESOURCE DETAIL screen for the selected resource. In input mode, the DELETE field appears where you can delete the resource detail entry.
PF12	Exit from the Administration menu.

See Also The appropriate *TIBCO Service Gateway* manual for proper settings for intermediate rollback.

Node Name List Screen

Function — The NODE NAME LIST displays the node names of the available Data Object Brokers.

To Display
- From the RESOURCE TYPE LIST screen, press PF4.
- From the RESOURCE DETAIL screen, press PF5.

Illustration of Node Name List Screen

```
S6BADM35   S6H1DOBB                NODE NAME LIST                2006NOV29 14:45:32

   AMDSDCPHURON          AMDSDCTHUR31          AMDSDCTHUR32          AMDSDCTHUR40
   AMDSDCTHUR41          CASPER                CASPER00              GEORGETOWN
   V4R1




     ENTER-OUTBOUND DETAIL  PF2-INBOUND DETAIL  PF4-TYPE LIST  PF5-PROCESS
NO MORE DATA TO BE DISPLAYED
```

Key Commands

Enter	Display RESOURCE DETAIL screen for outbound paths (HRN resource type). Place your cursor beside the desired NODE NAME and press Enter.
PF1	Display help information about the current screen.
PF2	Display RESOURCE DETAIL screen for inbound paths (HIN resource type). Place your cursor beside the desired NODE NAME and press PF2.

PF3	Return to the main menu.
PF4	Display the RESOURCE TYPE LIST screen.
PF5	Access the selected node All subsequent commands in the Administration menu apply to the chosen node until you exit from the menu or use the NODE NAME LIST option to choose another node. The local node appears by default whenever you select the NODENAME LIST option, regardless of any previous node processing. **Note** Remote administration is supported only for Data Object Brokers at the same release level.
PF12	Exit from the Administration menu.

Path List Screen

Function — Lists all the connected paths within the resource identified in the screen title. It also shows the job name under which the Execution Environment is running, the status of the path, and its utilization.

To Display — From the RESOURCE DETAIL screen, press Enter.

Illustration of the Path List Screen

```
S6BADM34   S6H1DOBB          PATH LIST FOR API DEFAULT0       2006NOV29 14:52:04

            CURRENT    JOB              LAST    TIMES      MESSAGE            LAST
      ID     USER      NAME     STATUS MSG      USED        COUNT PARTNER    USER
  T00E6000                      IDLE               0            0 U00E6000
  T00E6001                      IDLE               0            0 U00E6001
  T00E6002                      IDLE               0            0 U00E6002
  T00E6003                      IDLE               0            0 U00E6003
  T00E6004                      IDLE               0            0 U00E6004
  T00E6005                      IDLE               0            0 U00E6005
  T00E6006                      IDLE               0            0 U00E6006
  T00E6007                      IDLE               0            0 U00E6007
  T00E6008                      IDLE               0            0 U00E6008
  T00E6009                      IDLE               0            0 U00E6009




   ENTER-REFRESH  PF2-TYPE LIST  PF4-GROUP LIST  PF5-DETAIL  PF11-STOP SERVER
 NO MORE DATA TO BE DISPLAYED
```

Key Commands

Enter	Refresh the display.
PF1	Display help information about the current screen.
PF2	Display the RESOURCE TYPE LIST screen.
PF3	Return to the main Administration menu.
PF4	Display the RESOURCE GROUP LIST screen.

PF5	Return to the RESOURCE DETAIL screen for the current resource.
PF11	Stop a selected path. You are prompted to confirm the stop request.
PF12	Exit from the Administration menu.

Message Turnaround Time Screen

Function — Displays a summary of message response times for all resource types.

To Display — From the RESOURCE TYPE LIST screen, press PF9.

Illustration of the Message Turnaround Time Screen

```
S6BADM36  S6H1DOBB      MESSAGE TURNAROUND TIME BY TYPE       2006NOV29 15:01:42
                                                                    DELTA
     ALL                PERCENTAGE OF TOTAL MESSAGES
            128ms 256    512  1024  2048  4096  8192 >8192          TOTAL  DELTA
   ADA          0     0     0     0     0     0     0     0              0
   API          0     0     0     0     0     0     0     0              0
   DAT          0     0     0     0     0     0     0     0              0
   DB2          0     0     0     0     0     0     0     0              0
   HRN          0     0     0     0     0     0     0     0              0
   IDM         19    15     6    10    19    15     6    10             48
   IMS          0     0     0     0     0     0     0     0              0


    ENTER-REFRESH  PF2-ONLINE ONLY  PF4-COMMON ONLY  PF5-COMBINED  PF9-TYPE LIST
 NO MORE DATA TO BE DISPLAYED
```

Understanding Message Turnaround Times

The Message Turnaround Time By Type screen presents a matrix by resource type of the message turnaround times. Each column shows the percentage of messages for that resource that fall in the millisecond range indicated.

At the right, the two columns show the total number of messages for that resource and the change in number of messages since the last refresh. When the screen is refreshed, the time elapsed since the previous refresh could appear beside the word “DELTA” in the top right corner of the screen.

The data displayed is indicated by the value at the top left corner of the matrix:

Report Type	Pertains to data from
ONLINE	Online paths only, available to terminal-based transactions that tend to run at a higher priority.
COMMON	General access, used perhaps by batch or terminal-based transactions, excluding values from online-only paths.
ALL	Contains both the ONLINE and COMMON accumulated values. This is the default display.

Key Commands

Enter	Refresh the display.
PF1	Display help information about the current screen.
PF2	Display online connections only.
PF3	Return to the main Administration menu.
PF4	Display common connections only.
PF5	Display all connections. This is the default.
PF9	Return to the RESOURCE TYPE LIST screen.
PF12	Exit from the Administration menu.

Defined Schedule List Screen

Function — Use the DEFINED SCHEDULE LIST screen to view a list of existing schedules. From this screen you can select a schedule to edit, or define new schedules. You can display the resource details of a schedule and the resources associated with it. Normally for most resources, you should not need more than two or three schedules.

To Display — From the RESOURCE TYPE LIST screen or the RESOURCE DETAIL screen, press PF10.

Illustration of the Defined Schedule List Screen

```
S6BADM38   S6H1DOBB           DEFINED SCHEDULE LIST           2006NOV29 15:02:58

     COMMON               SCHEDULE1




 ENTER-SCHEDULE  PF2-DETAIL  PF4-TYPE LIST  PF5-NEW SCHEDULE  PF9-ASSOCIATES
 DISPLAY FROM TOP OF DATA
```

Key Commands

Enter	Display the resource schedule for the selected schedule name.
PF1	Display help information about the current screen.
PF2	Display the RESOURCE DETAIL for the selected schedule.

PF3	Return to the main Administration menu.
PF4	Display the RESOURCE TYPE LIST screens.
PF5	Define a new schedule.
PF9	Display the resources associated with the selected schedule.
PF12	Exit from the Administration menu.

Resource Schedule Screen

Function

Use the RESOURCE SCHEDULE screen to view the schedule for a specific resource in the same format as it is presented on the RESOURCE DETAIL SCREEN. Create a copy of the schedule, create new schedules, or delete or update existing schedules from this screen.

To Display

- From the DEFINED SCHEDULE LIST screen, select a schedule name and press ENTER.
- From the USERS OF SCHEDULE screen, press PF2.

Illustration of the Resource Schedule Screen

```
S6BADM37  S6H1DOBB      RESOURCE SCHEDULE COMMON              2006NOV29 15:04:40

 ACT        APPLICABLE DAYS          EXCEPTION    START   ONLINE   CONNECTIONS
     MON TUE WED THR FRI SAT SUN      DATE         TIME    ONLY        MAX
      Y   Y   Y   Y   Y   Y   Y                   00:00     N           10




                        TARGET RESOURCE ___ _________
ENTER-DETAIL PF2-SCH IX PF4-COPY PF5-ASSOCIATE PF9-NEW PF10-UPDATE PF11-DELETE
DISPLAY FROM TOP OF DATA
```

Key Commands

Enter	Display the Resource Detail for the schedule.
PF1	Display help information about the current screen.
PF2	Display the DEFINED SCHEDULE LIST screen.

PF3	Return to the main Administration menu.
PF4	Copy the schedule to a new schedule.
PF5	Associate the schedule with a named resource.
PF9	Create a new schedule.
PF10	Update the displayed schedule.
PF11	Delete the displayed schedule when it is no longer needed.
PF12	Exit from the Administration menu.

How to Use Schedules

Resources are scheduled based on when they are available and how many of them are to be made available. Normally, each resource has only one or two schedule entries associated with it.

On the RESOURCE SCHEDULE screen, use the **Target Resource** field at the bottom of the screen when you copy a schedule or when associating a schedule with a resource group.

Using Exception Entries

When you define exception dates for a schedule, any exception records are automatically removed from the schedule file at the end of the day they expire. Dates are in the format *CCYYMMMDDD*, and an asterisk (*) indicates the current day. Exception entries always override the normal settings for the schedule. A typical schedule consists of one line showing the normal settings, and one or more lines for days when the normal settings do not apply.

To set up a new schedule:

1. Select a resource group and press PF10 to see a list of available schedules.

   The DEFINED SCHEDULE LIST appears.

2. Press PF5 to display an empty resource schedule screen in input mode.

   You can also press PF9 from the RESOURCE SCHEDULE SCREEN to display an empty resource schedule screen in input mode.

3. Complete as many lines as you need and press PF10 to save the new schedule.

   The first line of the new schedule should be the normal settings for that schedule. Subsequent lines define exceptions to or variations on this schedule.

4. Press PF5 to associate the schedule with a resource group.

Illustration of the Resource Schedule Screen in Input Mode

```
S6BADM37  S6H1DOBB      RESOURCE SCHEDULE COMMON              2006NOV29 15:04:40

 ACT        APPLICABLE DAYS          EXCEPTION    START  ONLINE  CONNECTIONS
      MON TUE WED THR FRI SAT SUN      DATE         TIME   ONLY       MAX
       Y   Y   Y   Y   Y   Y   Y     __________   00:00    N          10
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___
   _   _   _   _   _   _   _   _     __________   __:__    _          ___

                          TARGET RESOURCE ___ _________
 ENTER-DETAIL PF2-SCH IX PF4-COPY PF5-ASSOCIATE PF9-NEW PF10-UPDATE PF11-DELETE
 DISPLAY FROM TOP OF DATA
```

Users of Schedule Screen

Function — Use this screen to display a list of all resources that use a specified schedule.

To Display — From the DEFINED SCHEDULE LIST screen, press PF9.

Illustration of the Users of Schedule Screen

```
S6BADM39   S6H1DOBB      USERS OF SCHEDULE COMMON              2006NOV29 15:07:13

     ADAADAFSLV1          ADATESTC             APIDEFAULT0          APIFILEGATE
     APIGEORGETN          APILESSMITH          DATDATACOM0          DATDATACOM1
     DATIMPSRVR           DB2DB2PUT01          DB2DB2PUT04          DB2DB2SRVR
     DB2DB2SRVR2          DB2DCOM2DB2          DB2RFSDB21           DB2RFSDB22
     HINCASPER            HINCASPER00          HINGEORGETN          IDMRFSIDMS
     IDMRFSIDMS1          IMSIMSDRA            IMSRFSIMS            IMSRFSIMS41









         ENTER-DETAIL  PF2-SCHEDULE  PF4-TYPE LIST  PF5-SCHEDULE INDEX
 DISPLAY FROM TOP OF DATA
```

Key Commands

Enter	Display the RESOURCE DETAIL screen for the named schedule.
PF1	Display help information about the current screen.
PF2	Display the RESOURCE SCHEDULE for the named schedule.
PF3	Return to the main Administration menu.
PF4	Display the RESOURCE TYPE LIST screen.

PF5	Display the DEFINED SCHEDULE LIST screen.
PF12	Exit from the Administration menu.

Monitoring with z/OS System Management Facility (SMF)

SMF accounting and performance data can optionally be collected from both Execution and Data Object Broker Environments. Collected data will be written out to the z/OS system SMF data sets for subsequent processing and analysis. Collection of SMF data is controlled by parameter values in both environments. Each environment is treated independently, although where applicable, units of related work processed in an EE or DOB can be correlated using common unit of work identifiers. See *TIBCO Object Service Broker Monitoring Performance* for more information on processing and analyzing SMF data and *TIBCO Object Service Broker Parameters* for information on the specific values and function of SMF-related parameters.

JCL(USERMODE) can be used to set the SMF Subsystem ID that is recorded on all SMF Object Service Broker records. This identifier is useful wen extracting out SMF data for TIBCO Object Service Broker. The default is S6B but USERMODE allows you to use another value (such as HRN) for compatibility with older TIBCO Object Service Broker releases and site specific reporting procedures. For details, see Customizing the Message Prefix and SMF Subsystem Identifier, page 48.

SMF Object Service Broker data can be extracted from the z/OS SMF data sets by using either the supplied S6BSMFEX utility or the z/OS system utility IFASMFDP. Refer to *TIBCO Object Service Broker for z/OS Utilities* for information on S6BSMFEX and other SMF utilities that can be used to manipulate and report SMF data. If your SMF data is in VSAM format, you will need to use IFASMFDP to extract the data to a sequential file.

Enabling SMF Data Collection in the Data Object Broker

Setting the DOB parameter SMFRECORD to a value between 128 and 255 will enable DOB SMF recording. You might use the same the SMF number for your EE in order to facilitate the extraction of SMF data for an entire TIBCO Object Service Broker system. Setting the SMF record number can be done through either the //INITPARM parameter input file, or dynamically using the operator command:

```
MODIFY S6ELDOBA,PARM=SMFRECORD=217
```

The type of data recorded to SMF, distinguished by the SMF record sub-type number, is controlled by the following DOB parameters:

- SMF13INTERVAL
- SMF22INTERVAL
- SMF23INTERVAL

- SMF24INTERVAL
- SMF25INTERVAL
- SMF26INTERVAL
- SMF47INTERVAL

Enabling SMF Data Collection in the Execution Environment

Setting the EE parameter SMFTYPE to a value between 128 and 255 will enable EE SMF recording. Consider using the same SMF record value in both your DOB and EE in order to facilitate the extraction of SMF data for an entire TIBCO Object Service Broker system. Set the parameter values you wish to use for your EE in the CNTL(PARMxxxx) members corresponding to your EE configurations, then submit JCL(EECONFIG) to instantiate the values for execution. The amount and type of data recorded to SMF is controlled by the following EE parameters:

- SMFDETAIL
- SMFPERFORMANCE

Appendix A Installation Variables

This appendix describes the installation variables for TIBCO Object Service Broker and the TIBCO Service Gateways.

Topics

Modifying Installation Variables

The distribution data sets provided by TIBCO Software are installed to <HLQ>.*. During the installation process, they are copied to the live data sets listed below under <HLQNONV>.<INSTVER>.* or <HLQNONV>.<SLQ>.*, where they are customized using OSEMOD (found in the <HLQ>.FILECLS or <HLQNONV>.<INSTVER>.CLIST data set.

Distribution Data Set	Live Data Set
<HLQ>.OSB.FILEI	<HLQNONV>.<INSTVER>.OSB.JOBS
<HLQ>.FILECTL	<HLQNONV>.<INSTVER>.CNTL
<HLQ>.FILECLS	<HLQNONV>.<INSTVER>.CLIST
<HLQ>.FILEJCL	<HLQNONV>.<INSTVER>.JCL
<HLQ>.FILEMIG	<HLQNONV>.<INSTVER>.MIGRATE
<HLQ>.FILECTL(RELAYCFG)	<HLQNONV>.<SLQ>.RELAYCFG

If you must make any further modifications to customizable variables:

1. Make a backup copy of OSEMOD.
2. Alter OSEMOD variables following the guidelines in the following sections and save OSEMOD in a data set that is in your SYSPROC concatenation.
3. Copy the members that you are changing from the distribution data sets to the corresponding live data sets.
4. Run the OSEMOD edit macro against the live data sets.

Warning Do not change the original distribution data sets.

Required Variables

This section describes the primary installation variables.

Variable	Description	Default	Recommend using default
$ACLSEG$	Designated segment number where you want the audit log data table moved. This segment must be defined in the DBDLIB and must be available online at all times; see DBDLIB on page 442.	99	Y
$ACTCDE$	Accounting information to be used in installation JOB cards. The parameter string is placed inside the JOB card accounting field, for example: `//jobname JOB ($ACTCDE$)...`	/	
$ASMBLR$	Level of assembler to use. Compile options are specified in *$PARMASM$*.	ASMA90	Y
$BAKGDG$	Number of generation data group data sets to be kept for system backups.	255	Y
$BATQNM$	Initial TIBCO Object Service Broker batch server queue name. Refer to Step 2: Define an Initial Batch Queue on page 64.	ADMIN	Y
$BATSRI$	Member name in `<HLQ>.FILECTL` that contains the startup parameters for the batch server created during the installation process. Refer to Step 3: Prepare the Initial Batch Server on page 65.	XBATCHL1	Y
$BATUSR$	Initial user ID for the TIBCO Object Service Broker batch server created during the installation process. The user ID specified must be defined before the batch server is used.	BATSRV1	Y
$CEELIB$	High-level qualifier for the name of the Language Environment library.	CEE	
$CSSLIB$	High-level qualifier for the name of the IBM library that contains token routines used for locating MSBs properly.To find this value, use ISPF 3.4 with a Dsname Level of **.*CSSLIB*.	SYS1	

Variable	Description	Default	Recommend using default
$DMPGDG$	Number of generation data group data sets to be kept for the Data Object Broker. Used on the SYSMDUMP DD card.	5	Y
$DOBPORT$	The TCP/IP port number in the HCS relay parameter file (member RELAYCFG in <HLQ>.OSB.FILEI) that defines a z/OS Data Object Broker peer connection to a Windows Data Object Broker using TCP/IP.	9000	
$EENAME$	Name of the Native Execution Environment (NEE) to be used to run the session. This is the VTAM LU name or the communications ID used to log in to the Native Execution Environment. This value must be uppercase and at most 8 characters long. It also defines the node name of the z/OS NEE used for the TIBCO Object Service Broker UI connectivity in Windows.	OSBNEE	
$EEPORT$	The TCP/IP port number in the HCS relay parameter file (member RELAYCFG in <HLQ>.OSB.FILEI) that defines a z/OS Native Execution Environment for TIBCO Object Service Broker UI in Windows using TCP/IP.	9001	
$GDGUNT$	DASD device type for generation data group bases.	3390	Y
HLQ	High-level qualifier for data sets designated as distribution libraries, uploaded from TIBCO's download site.	OSB	
$HLQNONV$	High-level qualifier for non-VSAM data sets used by TIBCO Object Service Broker. The final data set name generated must be valid to z/OS.	OSB	
$HLQVSAM$	High-level qualifier for VSAM data sets used by TIBCO Object Service Broker. The final data set name generated must be valid to z/OS. You can use the same high-level qualifier for HLQ, *$HLQNONV$* and *$HLQVSAM$*.	OSB	
$HOST$	The symbolic name of the TCP/IP host system where the Data Object Broker resides (member RELAYCFG in <HLQ>.OSB.FILEI).	ZOS.OSB.COM	

Variable	Description	Default	Recommend using default
$INSTUNT$	DASD device type for allocating TIBCO Object Service Broker installation data sets.	3390	Y
$INSTVER$	Second-level qualifier for installation related data sets.	INS60	Y
$INSTVOL$	DASD volume for allocating installation data sets.	OSBS00	
$JESPRT$	JES2/JES3 print destination (for the SNAPR Data Object Broker parameter).	LOCAL	
$JOBNME$	Default jobname for installation JCL JOB cards. This value must be uppercase and at most 7 characters long.	INSTALL	
$JRNLSIZ$	The size of the journals in number of disk cylinders. For a minimal TIBCO Object Service Broker system accessing external database gateways, use a value of 25.	25	
$JSRGDG$	Number of journal spin accumulation generation data group data sets; for example, JOURNAL.SPINOUT. Use this parameter in conjunction with *$SPINLIM$* to control how often the journal spin generation data groups are merged by SPINMRG.	255	Y
MDL	Pattern used for selecting the Execution Environment communications identifier. See Determine Names of Execution Environment VTAM Application Definitions on page 413 for coding guidelines.	OSB9999	Y
$MQMLIB$	The high-level qualifier for the name of the WebSphere MQ library. To find this value, use ISPF 3.4 with a Dsname Level of **.SCSQLOAD.	MQM	
$MSGCLS$	Default message class for installation JCL JOB cards.	A	
$NDATCLAS$	Your site's defined SMS DATACLASS for non-VSAM data sets, as required.	STANDARD	

Variable	Description	Default	Recommend using default
$NMGTCLAS$	Your site's defined SMS MANAGEMENTCLASS for non-VSAM data sets, as required.	STANDARD	
$NODNAM$	Assigns a unique Data Object Broker name in a distributed data environment. The name can be from 1 to 16 characters. The name specified is for use in remote nodes to access data in the local node. A value must be specified.	DOB60ZOS	
$NOTIFY$	Default NOTIFY value for the installation JCL JOB cards. For example, you can use your TSO ID or the value of the &SYSUID variable.	&SYSUID	Y
$NSTOCLAS$	Your site's defined SMS STORAGECLASS for non-VSAM data sets, as required.	S6BNONV	
$OSBVOL$	VOLSER of DASD volume for allocating Object Service Broker data sets.	OSBS01	
$PRCCLS$	Default execution CLASS value for the installation JCL JOB cards.	A	
$REDODUP$	A conditional variable to indicate whether duplex redologs are to be implemented. To implement duplex redologs, specify Y. IF Y is specified, the default value of *$REDOVDF$* must be changed.	N	Y
$REDOSIZ$	The size of the redolog in number of disk cylinders. For a minimal system, use a value of 50.	50	Y
$REDOVDF$	Member name in <HLQ>.FILECTL that supplies IDCAMS statements for job S6A3ALOC in <HLQ>.OSB.FILEI to define the redolog data sets. If duplexing the redolog, specify DPLXREDO.	SPLXREDO	Y
$SEG0NAM$	Second-level qualifier for segment 0 page data sets. Use a name that is meaningful and unique.	SEG00	Y
$SEG0SIZ$	Size in cylinders for each MetaStor page data set. Use a minimum of 50.	100	Y

Variable	Description	Default	Recommend using default
$SEG1NAM$	Second-level qualifier for segment 1 page data sets. Use a name that is meaningful and unique.	SEG01	Y
$SEG1SIZ$	Size in cylinders for each segment 1 page data set. Use a minimum of 14.	25	Y
$SEG99NAM$	Second-level qualifier for page data sets on the segment specified by *$ACLSEG$* (default: segment 99). Use a name that is meaningful and unique.	SEG99	Y
$SEG99SIZ$	Size in cylinders for page data sets on the segment specified by *$ACLSEG$* (default: segment 99). For a minimal system, use a value of at least 14.	50	Y
$SERVERID$	Identifier to be used to run the server sessions. This value corresponds to the Resource Management Group value within a specific TYPE. Refer to Resource Detail Screen on page 370. This value must be uppercase and at most 8 characters long.	DEFAULT0	Y
SLQ	Second-level qualifier for installation data sets.	OSB60	Y
$SNPCLS$	SYSOUT class for snap dump output for Data Object Broker parameter SNAPC.	A	Y
$SPINLIM$	Number of journal spin generation data group data sets created before they are merged by SPINMRG into a single data set. Default must not be changed until SPIN verification is complete in post-installation. This variable is used in conjunction with *$JSRGDG$*.	2	Y
$SVCNUM$	Type-3 TIBCO Object Service Broker SVC number assigned to be used in establishing an authorized Execution Environment. The number must be between 200 and 255.	0	
$SYSPRT$	SYSOUT class for installation jobs output. Default directs SYSOUT to the same class as the default message class (*$MSGCLS$*) on the JOB card.	*	Y

Variable	Description	Default	Recommend using default
TDS	Communications identifier assigned to the Data Object Broker. It is required by the Execution Environments to establish communications with a Data Object Broker.	OSBDOB	Y
$VDATCLAS$	Your site's defined SMS DATACLASS for VSAM data sets, as required.	STANDARD	
$VMGTCLAS$	Your site's defined SMS MANAGEMENTCLASS for VSAM data sets, as required.	STANDARD	
$VSTOCLAS$	Your site's defined SMS STORAGECLASS for VSAM data sets, as required.	S6BVSAM	

SMP/E Variables

Variable	Description	Default	Recommend using default
$PARMASM$	Assembler compile options for the level of assembler specified in *$ASMBLR$.*	LIST,XREF(SHORT), ALIGN,DECK, NOOBJECT	Y
SMP	Second level qualifier used to allocate all libraries related to SMP/E processing.	SMP60	
$SMPPROC$	Name of your site's SMP/E procedure. For the automated install option, this procedure is provided in <HLQ>.OSB.FILEI and the default name must be SMPPROC.	SMPPROC	Y

Migration Variables

Variable	Description	Default	Recommend using default
$MIGRPSW$	Password for migration.	SYSADMIN	Y
$MIGRUSR$	User ID for migration.	SYSADMIN	Y

Service Gateway Variables

This section describes the variables for the TIBCO Service Gateway components. For additional information about Service Gateways, see the *TIBCO Service Gateway* manuals.

Adabas

Variable	Description	Default	Recommend using default
AD	TIBCO Object Service Broker Fail Safe transaction database name.	@ADAFSTRXDB	Y
$ADAFSL$	Fail Safe level. Value is 0 (off) or 1 (on).	0	Y
$ADAID$	The TIBCO Object Service Broker level-1 user ID that the Gateway uses to log in to the Data Object Broker.	ADA01	
$ADAJNM$	The job name for the Adabas Service Gateway.	ADAGWY	
$ADALOD$	The Adabas load library.	SAG.ADA744.LOADLIB	
$ADANUM$	The number of gateway subtasks.	1	
$ADASRV$	The TIBCO Object Service Broker server ID for the Adabas Service Gateway. The corresponding Resource Management group entry must match this name in the Data Object Broker for the ADA server type.	ADASRV	Y

CA-Datacom

Variable	Description	Default	Recommend using default
$CADATA$	CA-Datacom Data Dictionary object library. Must be a fully qualified data set name.	CAI.DCOM112.CADB0LLD	
$CAIPC$	CA's IPC object library. Must be a fully qualified data set name.	CAI.CCS300.CAILIB	
$CAILIB$	CA's ILIB object library. Must be a fully qualified data set name.	CAI.CCS300.CAILIB	
$CAMACL$	CA-Datacom macro library. Must be a fully qualified data set name.	CAI.DCOM112.CAIMAC	
D	TIBCO Object Service Broker Fail Safe database number.	00888	Y
$DCOMCXX$	Data Dictionary C*xx* data set. Must be a fully qualified data set name.	CAI.DCOM112.CXX	
$DCOMDID$	Data Dictionary user ID.	DATACOM-INSTALL	Y
$DCOMDPW$	Data Dictionary password.	NEWUSER	Y
$DCOMFSL$	Fail Safe level. The Value is 0 (off) or 1 (on).	0	Y
$DCOMID$	TIBCO Object Service Broker user ID that the Gateway uses to log in to the Data Object Broker.	SYSADMIN	
$DCOMJNM$	Job name for the Service Gateway for Datacom.	DATGWY	
$DCOMLOD$	CA-Datacom load library. Must be a fully qualified data set name.	CAI.DCOM112.CAILIB	
$DCOMNUM$	Number of gateway subtasks.	3	
$DCOMSRV$	TIBCO Object Service Broker server ID for Service Gateway for Datacom. The corresponding Resource Management group entry must match this name in the Data Object Broker for the DAT server type.	DEFAULT	

CICS Variables

This section describes the CICS installation variables.

Variable	Description	Default	Recommend using default
$AAPPLID$	CICS APPLID for the Application-Owning Region (AOR).	CICSAOR	
$CICSMAC$	CICS macro data set.	CICSTS31.CICS.SDFHMAC	
$DFHCSD$	CICS DFHCSD filename.	CICSTS31.CICS.DFHCSD	
$OSTARAOR$	Resource Definition Online LIST name for resources defined to the CICS AOR.	OSBAOR	Y
$OSTARNMR$	Resource Definition Online LIST name for resources defined to the CICS standalone region.	OSBNMR	Y
$OSTARTOR$	Resource Definition Online LIST name for resources defined to the CICS Terminal Owning Region (TOR).	OSBTOR	Y
$SDFHLOAD$	CICS SDFHLOAD data set.	CICSTS31.CICS.SDFHLOAD	
$TAPPLID$	CICS APPLID for the TOR.	CICSTOR	
$XAOR$	Remote CICS system name for the AOR. This name is used to define sample MRO CONNECTIONs between the TOR and the AOR.	XAOR	
$XTOR$	Remote CICS system name for the TOR. This name is used to define sample MRO CONNECTIONs between the AOR and the TOR.	XTOR	

DB2

Variable	Description	Default	Recommend using default
$DB2NAME$	A unique jobname for the DB2 Gateway as shown in the CNTL member DB2BATCH.	OSBDB2	
$DB2SSI$	DB2 subsystem with which you want the TIBCO Object Service Broker server to communicate.	DSN8	
$PLNAME$	Plan name of the TIBCO Object Service Broker Gateway for DB2 data.	OSBDB2S	
$SSQLASM$	Source of generated static SQL modules. It must be a fully qualified data set name.	SSQLASM	
$SSQLBND$	Partitioned data set used to hold include cards for binding the Gateway with static SQL. It must be a fully qualified data set name.	SSQLBND	
$SSQLDBRM$	DBRM library for static SQL modules. It must be a fully qualified data set name.	SSQLDBRM	
$SSQLLIST$	Source listing for generated static SQL modules. It must be a fully qualified data set name.	SSQLLIST	
$SSQLOAD$	Load library for static SQL handlers. It must be a fully qualified data set name.	SSQLLOAD	
$SSQLOBJ$	Object code for generated static SQL modules. It must be a fully qualified data set name.	SSQLOBJ	
$SYSDB2$	DB2 load library. It must be a fully qualified data set name.	DSN810.SDSNLOAD	

IDMS/DB

Variable	Description	Default	Recommend using default
$IDMSCTL$	The CA-IDMS system control data set (SYSCTL). It must be a fully qualified data set name.	IDMS.R150.SYSCTL	
$IDMSDIC$	The CA-IDMS dictionary name.	APPLDICT	
$IDMSDMCL$	The CA-IDMS DMCL.	IDMSDMCL	
$IDMSFSLV$	The IDMS/DB Fail Safe level.	0	Y
$IDMSJNM$	The job name for the Service Gateway for IDMS/DB.	IDMGWY	
$IDMSLIB$	The name of the CA-IDMS load library. This parameter is required if you are installing the Service Gateway for IDMS/DB. It must be a fully qualified data set name.	CAI.IDMS.R150.LOADLIB	
$IDMSLV7P$	The IDMS/DB level-7 TIBCO Object Service Broker password.	SYSADMIN	Y
$IDMSLV7U$	The IDMS/DB level-7 TIBCO Object Service Broker user ID.	SYSADMIN	Y
$IDMSNUM$	The number of gateway subtasks.	3	
$IDMSSCH$	The CA-IDMS subschema name.	EMPSS01	
$IDMSSRV$	The TIBCO Object Service Broker server ID for the Service Gateway for IDMS/DB. The corresponding Resource Management group entry must match this name in the Data Object Broker for the IDM server type.	DEFAULT	Y
$IDMSXTR$	The high-level qualifier for the TIBCO Object Service Broker CA-IDMS extract utility data sets.	OSB.OSB60	

IMS/DB

Variable	Description	Default	Recommend using default
$DRASUFX$	Suffix of the DFSPZP*nn* module of the IMS/DRA interface.	00	
$IMSDBD$	Name of the IMS/DB DBD library. It must be a fully qualified data set name.	IMS1010.IMS1.DBDLIB	
$IMSID$	The IMS subsystem ID to which the IM2 or IMS gateway will connect. It can be of maximum length 4 characters.	I10E	
$IMSIVPDB$	Name of the IMS IVP database.	IMS1010.IMS1.DFSIVD1	
$IMSIVPIX$	Name of the index for the IMS IVP database	IMS1010.IMS1.DFSIVD1I	
$IMSJNM$	The job name for the IMS/DB Service Gateway.	IMSGWY	
$IMSMAC$	Name of the IMS macro library. It must be a fully qualified data set name.	IMS1010.SDFSMAC	
$IMSPSB$	Name of the IMS PSB library. It must be a fully qualified data set name.	IMS1010.IMS1.PSBLIB	
$NUMSRVR$	Number of Gateways to start. Must be 1 for all server types *except* DRA.	3	Y
$PROCLB$	Name of the IMS proc library. It must be a fully qualified data set name.	IMS1010.IMS1.PROCLIB	
$PSBNAM$	IMS PSB name.		
$SDFSRESL$	Name of the IMS RESLIB load library. It must be a fully qualified data set name.	IMS1010.IMS1.SDFSRESL	

Miscellaneous Variables

This section describes miscellaneous installation variables.

Data Join/Split Conversion

Variable	Description	Default	Recommend using default
$PSW117$	The password for user *$USR117$*.	SYSADMIN	Y
$USR117$	A TIBCO Object Service Broker user ID authorized to perform a JOIN or SPLIT data conversion.	SYSADMIN	Y

Appendix B Configurations for Communications

This chapter describes how to configure your TIBCO Object Service Broker communications.

Topics

Overview

Determine the Communications Method

You can use the following communications services for communications between the Data Object Broker and other TIBCO Object Service Broker components, such as client processes, Execution Environments, external database gateways, and peer Data Object Brokers:

- VTAM (for terminals connecting to an Execution Environment only)
- Cross Memory Services (XMS)
- TCP/IP
- XCF

Each of the components that comprise TIBCO Object Service Broker can reside on different platforms. For details about setting up cross-platform communications using TCP/IP refer to Configuring TCP/IP on page 423. For details about configuring communications between peer TIBCO Object Service Broker systems residing on different platforms, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Performance Considerations

For best performance, we recommend Cross Memory Services. Cross Memory Services provides better performance than TCP/IP, requiring less processor utilization and resource assignment. TCP/IP is the only method if communications is across platforms.

Distribution of Processor Cycles

If you are running symmetric cross memory services, where both address spaces are nonswappable, the processing cycles are shared equally between the two address spaces. In contrast, if you are running asymmetric cross memory services, where one address space (the Execution Environment) is swappable, the processor cycles used for moving data are accounted to the user Execution Environments.

Configuring VTAM Communications

If you are using VTAM for terminals that connect to an Execution Environment, review this section with your site's VTAM systems programmer.

VTAM Application Definitions

A VTAM application program requires the exclusive use of a VTAM application definition. This definition must be stored and activated by VTAM, executing on the same machine as the VTAM application program. With the exception of the VTAM network names/ACB names assigned, VTAM application definitions can be defined identically.

General Implementation Suggestions

Consider the following implementation suggestions:

Make CLISTs/EXECs or JCL Independent of Domain

VTAM requires that the ACB name be unique within a domain but allows it to be duplicated within a network. This enables pools of Execution Environment VTAM application definitions in different domains to specify the same VTAM ACB names. On the other hand, VTAM requires that network names be unique within the network.

If you implement your VTAM network resources using the following steps, you can take advantage of this shareability:

1. Install Execution Environment VTAM application definition major nodes with both VTAM network names and ACB names defined.
2. Specify the same format of VTAM ACB names for VTAM application definition major nodes that are installed in different VTAM domains.
3. Use the model Execution Environment communication identifier as a pattern for the VTAM ACB name.

Using this technique, the TSO EXECs or JCL used to execute Execution Environments are not dependent on the VTAM domain where they execute. In an environment of multiple VTAM domains with shared DASD or shared SPOOL, this means that CLISTs/EXECs can be shared among domains. It also means that Execution Environments that execute in batch do not have to be submitted for execution within a particular domain.

Use Distinct Prefixes to Define Multiple Pools

The selection of a particular prefix defines the pool of VTAM application identifiers; multiple pools in the same VTAM domain can be defined by choosing distinct prefixes.

Use Separate VTAM Nodes

File Execution Environment VTAM application definitions in separate VTAM major nodes. If multiple pools of Execution Environment VTAM application definitions are being used, file a separate VTAM major node for each pool.

Sample VTAM Applications

Sample VTAM major node and logmode table definitions are included in the ASM data set. These definitions are provided to assist your VTAM systems programmer in building the definitions required by TIBCO Object Service Broker. The major nodes should be stored in SYS1.VTAMLST in all the VTAM domains within your network. The modifiable samples are distributed in the following member names:

SMPDRVAD	A VTAM major node containing VTAM application definitions for four Execution Environments.
SMPLGMDE	A VTAM logmode table definition. The table name produced is S6BNCMDT. Modifiable operands are identified by an at sign (@) character preceding comment text on the same statement record as the keyword operand. Procedures for modifying the sample VTAM applications are outlined below.

Create VTAM Application Definitions for Execution Environments

Execution Environment VTAM application definitions are in a pool available for the general use of all Execution Environments including native Execution Environments, regardless of the intended Data Object Broker. They are allocated to Execution Environments on a first-come, first-serve basis. For more information on defining keywords in the application definition, refer to VTAM Application Definition Keyword Operands on page 415.

Before making modifications to the sample VTAM application definitions used by the Execution Environment, you must determine the following:

- How many VTAM application definitions you need to create
- The names that are to be assigned to these VTAM application definitions

Determine the Number of Execution Environment Definitions

Within a given VTAM domain, more than one VTAM application definition is likely required for use by Execution Environments. Use the following criteria to determine how many VTAM application definitions you need to define in a single domain:

- Each TSO user logged in to TIBCO Object Service Broker requires a unique VTAM application definition for the duration of their session.
- Each Native Execution Environment requires a unique VTAM application definition.
- Factor in an additional 25% to accommodate growth.

Determine Names of Execution Environment VTAM Application Definitions

Each Execution Environment VTAM application definition specifies a unique VTAM network name and/or VTAM ACB name as required by VTAM. VTAM supports the use of either the VTAM network name or the ACB name as the VTAM application identifier requested when the VTAM application program OPENs its VTAM ACB.

How an Execution Environment selects VTAM Application Identifiers

An Execution Environment selects the VTAM application identifier to use from a pool containing a range of numerically suffixed identifiers. The format of these VTAM application identifiers must follow the model that is specified by the OSEMOD installation variable *MDL* when TIBCO Object Service Broker is installed. An Execution Environment selects the VTAM application identifier that matches the model and has the lowest numeric suffix not already in use.

To ensure that the entire pool of application identifiers is available, note that:

- The last consecutively sequenced application identifier completes the pool. VTAM application identifiers that are defined after a gap in the sequencing are not in the pool and are not available for selection by VTAM for use in Execution Environments.
- If an identifier within a pool is inactive to VTAM, and that identifier is not at the end of the pool, all identifiers remaining to the end of the pool are not accessible or usable.

Specify the Application Identifier Model

Specify the format of the application identifiers in the pool to the Execution Environments by coding the MDL parameter in the PARM*xxx* members in the CNTL data set during the installation process. It is defined by the OSEMOD installation variable *MDL*. Refer to Appendix A, Installation Variables, on page 393 for details about the OSEMOD installation variables. Refer to *TIBCO Object Service Broker Parameters* for additional information about the MDL Execution Environment parameter.

The first available application identifier that matches this model is selected from the pool. It is not necessary to define the model itself as a VTAM Network or ACB name.

The SMPDRVAD member in the ASM data set, described in OSB9999. OSB is the prefix that defines the pool; 9999 is the suffix that represents that a four digit number, starting at 0001, is to be used by the identifiers as they are assigned.

Define the Pool of VTAM Application Identifiers

Define the pool of VTAM application identifiers as follows:

- Each identifier must consist of a prefix and a suffix, and its total length must not exceed eight characters. The prefixes and suffixes for all VTAM application identifiers in a particular pool must have the same length.
- The prefix part of each identifier within a pool must be the same. It must be one to seven alphanumeric characters.

- The suffix part of the identifiers must be between one and seven numeric characters. The suffixes must be filled with zeros on the left, to the length of the suffix. The suffixes must be assigned starting at one, and increase consecutively by one, until the maximum number of identifiers for the pool are defined.

Sample VTAM Application for an Execution Environment

Use as a model the SMPDRVAD member in the ASM data set. It displays a single application definition for an Execution Environment. It also defines a pool of four VTAM application definitions, using both network and ACB names:

Network Name	ACB Name
AOSB0001	OSB0001
AOSB0002	OSB0002
AOSB0003	OSB0003
AOSB0004	OSB0004

Define a VTAM ACB Definition for Each Native Execution Environment

The Native Execution Environment requires a dedicated VTAM application definition. Since it is dedicated to the Execution Environment during execution, define a unique VTAM ACB definition for each Native Execution Environment.

Define Native Execution Environment application definition parameters the same as Execution Environment pool definitions, except for the following changes:

- Do not define it as part of the normal Execution Environment application definition pool, for example, OSB9999.
- Do not define it with a numeric value.

VTAM Application Definition Keyword Operands

Use the following keywords to define your VTAM applications, as described in the previous sections.

DLOGMOD	Specify the VTAM logmode table entry name that supplies an LU Profile 6.1. HNCLUP61 is supplied in the sample VTAM logmode table distributed with TIBCO Object Service Broker. See Assemble and Link-edit Logmode Table on page 418.

EAS	Estimated maximum number of VTAM sessions to be anchored in the TIBCO Object Service Broker region for some proportion (for example, 90%) of the time that the region is active. • In the case of a Data Object Broker, this is the maximum number of sessions from end users, external database gateways, and utility Execution Environments that the Data Object Broker can concurrently have active. • In the case of an Execution Environment, this is the maximum number of sessions that an Execution Environment can concurrently have active with its Data Object Broker.

Most external database gateways and the CICS Execution Environment can concurrently engage in more than one session with a Data Object Broker.

MAXPVT	The maximum amount of storage in the TIBCO Object Service Broker region's private area that VTAM can use for queueing inbound data. This parameter is meaningful only when there is no outstanding VTAM RECEIVE request. TIBCO Object Service Broker uses VTAM in such a way that there is always an outstanding VTAM RECEIVE request. Therefore, this parameter has no impact on TIBCO Object Service Broker operations.

MODETAB	The name of the VTAM logmode table to be associated with the VTAM application program LU. The logmode table is used only for sessions where the TIBCO Object Service Broker region takes the part of an SLU (Secondary Logical Unit). Provide the load module name, which is produced by assembling and link editing the sample VTAM logmode table as mentioned in Create TIBCO Object Service Broker VTAM Logmode Table Definitions on page 418. You should make any necessary modifications to the sample provided before assembling. If the sample VTAM logmode table is incorporated into an existing site logmode table, specify the name of the site's revised VTAM logmode table.

SONSCIP	Decide if VTAM is to schedule the VTAM application program's SCIP exit with received UNBIND request units for those sessions where the VTAM application program is acting as a PLU (Primary Logical Unit). Specify YES or NO. TIBCO Object Service Broker VTAM application programs are prepared to function with either option of this keyword operand. If SONSCIP=YES is specified, the TYPE code in the UNBIND request unit can provide more detailed information about the reason for the loss of the session.

SRBEXIT	Specify if VTAM is to schedule the execution of the VTAM application program's asynchronous exits in SRB mode (YES or NO).

VPACING	Specifies the number of SNA normal flow request units that can be received by the TIBCO Object Service Broker region on a single session before a pacing response is solicited by the sender. It controls the volume of data in transit and arriving in the TIBCO Object Service Broker region for each session established with the TIBCO Object Service Broker VTAM application program. The use of this keyword operand can be circumvented by specifying keywords with certain values in the VTAM logmode table associated with the VTAM application definition. In the distributed samples, the VTAM logmode table is coded so that the pacing counts are taken from the VPACING keyword operands in the VTAM application definitions. Choose the value specified for this keyword operand with consideration for the capability of the network and the setting of the MAXPVT keyword, described previously.

Create TIBCO Object Service Broker VTAM Logmode Table Definitions

Assemble and Link-edit Logmode Table

The member SMPLGMDE in the ASM data set contains a VTAM logmode table with general logmode table entries that specifies logical unit profiles for LU 6.1 and LU 6.2. To install a VTAM logmode table, it must be assembled and link-edited into the VTAMLIB data set. The installation procedures assume the VTAM logmode table to be installed produces a load module named OSBNCMDT in the VTAMLIB data set; this name is used in the distributed sample VTAM application definitions.

For more information on keywords, refer to Define Keywords in VTAM Logmode Table on page 418.

Make Logmode Table Entries Accessible to VTAM

After making the appropriate changes, make the logmode table entries accessible to VTAM in one of the following ways:

- Install the VTAM logmode table in the VTAMLIB data set.
- Incorporate the logmode table entries into VTAM logmode tables that already exist in the VTAMLIB data set.

Whichever method you use to make the logmode table entries accessible to VTAM, you must specify the name of the load module as the value of the MODETAB operands. You must also specify the name of the VTAM LU 6.1 logmode table entry as the value of the DLOGMOD operands for all VTAM application definitions that support Data Object Brokers or Execution Environments.

Define Keywords in VTAM Logmode Table

Use the following keywords to define the logmode table entries:

- COS, page 419
- ENCR, page 419
- LOGMODE, page 419
- PRIPROT, page 420
- PSNDPAC, page 420
- RUSIZES, page 420
- SECPROT, page 420

- SSNDPAC, page 421

COS

Specifies the name of the class of service that is to be used for this session. Class of service affects the selection of a virtual route for the session and the priority of service assigned to the movement of data on the selected virtual route.

The installation can make use of class of service to distribute network resources equitably among users with different performance requirements.

Assigning Different Routes and Transmission Priorities to Users

To assign different session routes and transmission priorities to different TIBCO Object Service Broker end users, you can do the following:

- Produce multiple VTAM logmode table entries, naming different classes of service for each entry.
- Assign a different VTAM logmode table entry to different TIBCO Object Service Broker VTAM application definition pools.

Execution Environments that select an available VTAM application definition from a particular pool are also selecting the virtual route and transmission priority associated with the class of service. Using this technique, for example, production TIBCO Object Service Broker end users could be granted the use of network resources over a different route and/or at a higher priority than test/development TIBCO Object Service Broker end users.

ENCR	Specifies the level of cryptography that is to be used on the session. TIBCO Object Service Broker does not provide private cryptography. If desired and installed, VTAM cryptography can be selected.
LOGMODE	Specifies the name to be assigned to the VTAM logmode table entry. The distributed sample contains an entry named HNCLUP61, which corresponds to the value of the DLOGMOD keyword operands in the distributed sample VTAM application definitions. The names used are not significant. If a name is changed, it must correspond with one of the names used in the VTAM application definitions that are created to support the Data Object Broker and Execution Environments.

PRIPROT	Specifies the primary logical unit protocols. The only PLU Protocol that is not forced by TIBCO Object Service Broker is the “chain response protocol” (bits 2 and 3). The only acceptable options for the chain response protocol are “definite response” (B'..10....') and “definite and exception response” (B'..11...'). Any other specification is converted to “definite and exception response”. To achieve the highest communication efficiency, it is recommended that bits 2 and 3 of the PRIPROT be specified as “definite and exception response”. With this value, TIBCO Object Service Broker solicits definite responses only when protocols require it; otherwise, it solicits exception responses.
PSNDPAC	Specifies the primary send pacing count. To achieve configuration flexibility, it is recommended that a value of X'00' be specified for this keyword operand. This causes VTAM to use the value of the VPACING keyword for the SLU as the primary send pacing count.
RUSIZES	Specifies the maximum sizes of the SNA request units that are sent by the PLU and SLU. The format of the value for this keyword operand can be found in the appropriate VTAM manual. The values that can be specified depend on the limits of the network’s capability to handle SNA request units of various sizes. TIBCO Object Service Broker builds and transmits SNA request units within chains so that each SNA request unit is sized to its maximum according to the ability of the network to handle it, as specified by the value of this keyword operand. This yields fewer SNA request units per chain, improving communication performance. If this keyword operand is omitted or coded with a value of 0 for either request unit size, TIBCO Object Service Broker uses the maximum request unit size as set by SNA. If too large a value is specified or defaulted, transmitted SNA request units could be rejected with sense data of X'800A' by a network node along the communication route.
SECPROT	Specifies the secondary logical unit protocols. The only SLU protocol that is not forced by TIBCO Object Service Broker is the “chain response protocol” (bits 2 and 3). The only acceptable options for the chain response protocol are “definite response” (B'..10....') and “definite and exception response” (B'..11....'). Any other specification is converted to “definite and exception response”. To achieve the highest communication efficiency, it is recommended that bits 2 and 3 of the SECPROT be specified as “definite and exception response”. With this value, TIBCO Object Service Broker solicits definite responses only when protocols require it; otherwise, it solicits exception responses.

SSNDPAC	Specifies the secondary send pacing count. To achieve configuration flexibility, it is recommended that a value of X'01' be specified for this keyword operand. This causes VTAM to use the value of the VPACING keyword for the PLU as the secondary send pacing count.

Configuring Cross Memory Services

Review this section if you are using Cross Memory Services (XMS) to enable the Data Object Broker to communicate with the Execution Environments and/or external database gateways and/or other peer Data Object Broker regions.

Requirements for Cross Memory Services

- Cross Memory Services can only be used to connect Data Object Broker, Execution Environment or gateway jobs running in the same z/OS image.
- TIBCO Object Service Broker requires access to authorized z/OS functions and facilities to use Cross Memory Services.
- The use of TIBCO Object Service Broker cross memory services is limited to APF authorized address spaces.

Resource Manager Installation

To ensure complete TIBCO Object Service Broker XMS resource clean up after an abnormal termination, TIBCO Object Service Broker uses a z/OS resource manager module to deallocate and free up the resources used by the abending task. The resource manager load module name is in the form S6BRM*xxx*, where the suffix *xxx* is a three-digit number representing the compatibility level of the module. The actual name of the module can be determined by listing the contents of the TIBCO Object Service Broker load library after completing the installation process.

See Also Configuring Authorized Libraries on page 22 for more information on making the load library authorized.

Configuring TCP/IP

Review this section if you are using TCP/IP to communicate between TIBCO Object Service Broker components—client processes, Execution Environments, Data Object Brokers, and external database gateways—on a z/OS system and TIBCO Object Service Broker components on z/OS or non-z/OS platforms.

Requirements for TCP/IP

You must have the following in place in order to use TCP/IP:

- Access to authorized z/OS functions and facilities that uses TCP/IP.
- A TCP/IP stack compatible with the Sockets Extended Macro API of IBM's TCP/IP for MVS Version 3 Release 2 or higher.
- Access to sufficient concurrent TCP/IP sockets. Refer to the next section, Configuration of Resources for UNIX System Services, for details.
- An OMVS segment defined for each user ID associated with TIBCO Object Service Broker that will be accessing TCP/IP functions. This is required by z/OS UNIX System Services.

  Alternatively, you can set up RACF so that it can use a default OMVS segment for users that do not have a specific OMVS segment. This is described in the *RACF Security Administrator's Guide* in the section Using default OMVS segments in USER and GROUP profiles.

  For the corresponding information applicable to CA TSS or CA ACF environments, refer to the section Define a System Default UID and GID in *eTrust CA-Top Secret Security for z/OS Cookbook r9* or the section Defining a Default OMVS UID and GID in *eTrust CA-ACF2 Security for z/OS - Administrator Guide r9*.
- Properly configured relay file, member RELAYCFG to associate the TIBCO Object Service Broker communications identifier with the TCP/IP application addressing information. Refer to The Relay File on page 424 for details.

Configuration of Resources for UNIX System Services

So that TIBCO Object Service Broker can access sufficient TCP/IP sockets concurrently, set the MAXFILEPROC UNIX System Services parameter to an appropriate value. TIBCO recommend MAXFILEPROC=256 at minimum. This value may need to be set much higher than 256 for highly active systems with a large user base. The maximum number of sockets allowed is set initially by the MAXFILEPROC parameter in PARMLIB member BXPPRM*xx*. To dynamically set MAXFILEPROC, type the following z/OS system command:

```
SETOMVS MAXFILEPROC=nnnnnn
```

To check your current setting of MAXFILEPROC, type:

```
D OMVS,O
```

To display your system resource limits and high-water marks, type this console command:

```
D OMVS,LIMITS
```

The output will help you allocate sufficient resources to ensure successful execution of TIBCO Object Service Broker.

Set the z/OS system-wide parameter `MAXPROCSYS` to limit the number of processes that execute concurrently. An insufficient number might cause your application to suffer such failures as U1979/U07BB abends as you use more UNIX services.

As a guideline, every Execution Environment (EE) uses at least two UNIX processes, one for the main TCB and the other for the TCB, which controls the communications `RELAY`. Also consider these examples:

- If you set `TASKPOSIXNUM=1` or default the parameter to 1, that is another UNIX process for POSIX-compliant code.
- If you have set TCP/IP communications to be active, that is one more control process and several worker processes set by the `tcbnum` parameter in the `S6BRELAY` file.

The Relay File

The relay file, RELAYCFG, in the CNTL data set, contains information about TIBCO Object Service Broker components that use TCP/IP. It associates TCP/IP host names and port numbers with the TIBCO Object Service Broker communications identifiers that are used by these components running on any supported platform. The file is a text file in XML format that must be modified whenever changes to the TCP/IP environment are needed. Each component could have a separate relay file, or a common file could be shared across a number of components.

If XCF communications relay is deployed, TCP/IP parameters must be merged with the XCF parameters, and the combined parameters are contained in RELAYCFG in the CNTL data set. For details on XCF parameters, see Configuring XCF Communications, page 430.

The order of the relay parameters for each node name will be the order of selection for that node. If merged with XCF parameters, XCF will be considered before TCP/IP for nodes PRODZDOB and PRODZNEE; see Relay File Samples, page 425.

Run USERMODD in the JCL data set to customize the data set name of the relay file.

If you specify DSNAME=NULLFILE in USERMODD, this will disable TCP/IP access. In jobs and started tasks where you want to use TCP/IP, add an S6BRELAY DD statement pointing to the relay file.

If you specify a non-null relay file in a batch job, it is likely to have a short term region requirement at startup of over 64MB, as it runs the XML PARSER. This may cause jobs to fail. USERMODD with DSNAME=NULLFILE or an S6BRELAY DD DUMMY JCL statement removes this storage requirement.

The installation process for TIBCO Object Service Broker copies RELAYCFG to the data set $HLQNONV$.SLQ.RELAYCFG.

$HLQNONV$.SLQ.RELAYCFG contains your live TCP/IP information. If you need to make changes to your TCP/IP configuration, use the CNTL member RELAYCFG to make and verify your changes, then copy the new information to $HLQNONV$.SLQ.RELAYCFG.

To override the data set name set by USERMODD, add a DDNAME S6BRELAY to your TIBCO Object Service Broker component or any other z/OS components requiring TCP/IP communications. If this override is invalid during the component initialization, then the TCP/IP support is disabled until you provide a valid parameter file. Once the relay file has been processed during component initialization, it is freed.

The relay file consists of a set of protocol specific parameters followed by a directory that maps communications identifiers to protocol specific parameters.

Relay File Samples

The following is a sample the TCP/IP section of the HCS relay file:

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <tcpipparms tcbnum="3" maxtcbsockets="50" />
   <directory>
      <node name="PRODZDOB">
         <tcpip host="zos1.mydomain.com" service="emprec" />
      </node>
      <node name="PRODSDOB">
         <tcpip host="solaris5.mydomain.com" port="26360" />
      </node>
      <node name="PRODZEE">
         <tcpip host="zos1.mydomain.com" port="22636" />
      </node>
      <node name="TESTDOB">
         <tcpip host="168.192.0.101" port="26362"/>
```

```
      </node>
   </directory>
</relay>
```

The element and attribute names in the relay file are case sensitive.

Note that TIBCO Object Service Broker treats the value for the `name` attribute in the `node` element as though it were in uppercase. For example, `<node name="MixedDOB">` is treated the same as `<node name="MIXEDDOB">`.

The following is a sample of merged HCS relay configuration file for TCP/IP and XCF:

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <tcpipparms tcbnum="3" maxtcbsockets="50" />
   <xcfparms groupname='S6BOSB'/>
   <directory>
      <node name="PRODZDOB">
         <xcf/>
         <tcpip host="zos1.mydomain.com" service="emprec" />
      </node>
      <node name="PRODSDOB">
         <tcpip host="solaris5.mydomain.com" port="26360" />
      </node>
      <node name="PRODZEE">
         <xcf/>
         <tcpip host="zos1.mydomain.com" port="22636" />
      </node>
      <node name="TESTDOB">
         <tcpip host="168.192.0.101" port="26362"/>
      </node>
   </directory>
</relay>
```

TCP/IP Protocol Parameters for the Relay File

The *tcpipparms* element specify TCP/IP specific parameters. The parameter values are static for the life of your TIBCO Object Service Broker component.

tcbnum	The number of z/OS tasks to be started that will process socket specific events (default is 3 and minimum is 1).
maxtcbsockets	The maximum number of sockets that a TCP/IP relay task will handle (the default and the minimum is 50).

tcpname	The name of the TCP/IP address space that you want to connect to. If the name is not specified, the system derives a value from the TCP/IP system configuration file, as described in the IBM publication *z/OS Communications Server: IP Configuration Reference*.

The Relay File Directory

The relay file directory maps communications identifiers to protocol specific parameters. Each node element defines a communications identifier. The following are attributes that you can used to define the TCP/IP parameters for a node.

host	The IP host name or IP address for the node. Names are limited to 255 characters and IP addresses can be represented in IPV4 or IPV6 format.
port	The socket port number for the node. This is a number that can range from 1 to 32767.
keepalive	This optional value enables TCP/IP keepalive probes that will facilitate the detection of severed, idle connections, as well as possibly preventing the severing of idle connections by firewalls. The value is integer number between 1 and 65535; it specifies the interval in seconds between TCP/IP keepalive probes on outbound and inbound connections to this node.
service	The service name for the node that will be resolved to a port number. The names are limited to 255 characters.

Refer to the RELAYCFG member in the CNTL data set for sample TCP/IP configuration definitions.

Generating a Relay Utility

If you are upgrading from ObjectStar 4.1 or earlier to TIBCO Object Service Broker 5.x, you can use the S6BRLYGN utility to generate a relay file automatically. The utility converts the configuration data in a HRNPCSCM load module and creates the relay parameter file. The IN DD name specifies the load module library where HRNPCSCM is located job and the new configuration file is copied to the data set referenced by the OUT DD name. Any error messages are displayed in the file allocated to the SYSPRINT DD name.

```
//MYUSERGN JOB (0),'CONVERT PARMS',MSGCLASS=A,NOTIFY=MYUSER,TIME=10
//GENERATE EXEC PGM=S6BRLYGN,REGION=0M
//STEPLIB  DD DISP=SHR,DSN=$HLQNONV$.$INSTVER$.AUTH
//IN       DD DISP=SHR,DSN=SYS1.HRNCMCFG
//OUT      DD DISP=SHR,DSN=$HLQNONV$.$SLQ$.RELAYCFG
//SYSPRINT DD SYSOUT=*
```

Refer to the RELAYGEN member in the JCL data set for sample JCL to run the utility.

Dynamically Refreshing a Relay File

The relay parameters for a TIBCO Object Service Broker component can be refreshed by using a z/OS System modify command. The command allows an administrator to change the mapping of communication identifiers to TCP/IP hosts and ports. Changes to *tcpipparms* are not be honored and you must restart the component for the changes to become effective.

This command reloads the relay file allocated to the component.

```
F MYDOB,CS,REFRESH
```

This command loads the new relay file `$HLQNONV$.$SLQ$.NEWCFG`.

```
F MYDOB,CS,PARMDSN=$HLQNONV$.$SLQ$.NEWCFG
```

You can check the status of the relay by issuing the following command:

```
F MYDOB,CS,STATUS
```

The target of the relay commands need not be a Data Object Broker job as shown. Any z/OS Object Service Broker address space is eligible.

For CICS jobs, the format of the transaction is as follows:

```
HREL STATUS
```

Note that for CICS, the `CS,` sub-command prefix should not be used. The HREL transaction is intended to be used from a z/OS console. You may enter the transaction from a CICS terminal, but the response will be issued via a WTO and not back to your terminal. `STATUS` may be replaced by any valid relay command, such as `DISABLE`, `ENABLE`, `PARMDSN=` and `REFRESH`.

For additional information, see Chapter 16, Operator Commands.

The relay file is freed after the relay initialization is complete. So you can edit a relay parameter even though it is in use.

Verifying Relay File Syntax

The S6BRLYVA program allows the administrator of TIBCO Object Service Broker to verify that the syntax of a relay file is correct before starting a TIBCO Object Service Broker component that uses the file. The utility will parse the relay file allocated to the S6BRELAY DD name and will display any error messages in the SYSPRINT DD name.

```
//MYUSERGN JOB (0),'VERIFY PARMS',MSGCLASS=A,NOTIFY=MYUSER,TIME=10
//VALIDATE EXEC PGM=S6BRLYVA,REGION=0M
//STEPLIB  DD   DISP=SHR,DSN=$HLQNONV$.$INSTVER$.AUTH
//S6BRELAY DD   DISP=SHR,DSN=$HLQNONV$.$SLQ$.RELAYCFG
//SYSPRINT DD   SYSOUT=*
```

Refer to the RELAYVAL member in the JCL data set for sample JCL to run the utility.

Associated Documentation

For more information about IBM TCP/IP, refer to the appropriate IBM publication for the release of your z/OS system.

Configuring XCF Communications

Review this section if you are using Cross System Coupling Facility (XCF) services to communicate between TIBCO Object Service Broker components, where the Execution Environments and the external database gateways are running on different systems in a sysplex.

Requirements for XCF Communications

You must have the following in place in order to use XCF:

- XCF can be used only if the Execution Environment, the Data Object Broker, and the gateways are executing in the same sysplex.
- TIBCO Object Service Broker requires access to authorized z/OS functions and facilities to use XCF services.
- The use of TIBCO Object Service Broker XCF communications is limited to APF authorized address spaces.
- A properly configured relay file, to define XCF group name and member names for TIBCO Object Service Broker components.

The Relay File

RELAYXCF, in the CNTL data set, contains information about TIBCO Object Service Broker components that use XCF communications. This is a text file in XML format that defines the XCF group name and member names for TIBCO Object Service Broker components.

If TCP/IP relay is deployed, XCF parameters must be merged with the TCP/IP relay parameters, and the combined parameters are contained in RELAYCFG in the CNTL data set. For details on TCP/IP parameters, see Configuring TCP/IP, page 423.

The order of the relay parameters for each node name will be the order of selection for that node. If merged with TCP/IP parameters, XCF will be considered before TCP/IP for nodes PRODZDOB and PRODZNEE; see Relay File Samples, page 425. Run USERMODD in the JCL data set to customize the data set name of the relay file.

The installation process for TIBCO Object Service Broker copies RELAYCFG to the data set $HLQNONV$.SLQ.RELAYCFG. This data set contains your live XCF and TCP/IP information. If you need to make changes to your XCF configuration, use the CNTL member RELAYCFG to make and verify your changes, then copy the new information to $HLQNONV$.SLQ.RELAYCFG.

To override the data set name set by USERMODD, add a DDNAME S6BRELAY to your TIBCO Object Service Broker component. If this override is invalid during the component initialization, then the XCF support is disabled until you provide a valid parameter file. Once the relay file has been processed during component initialization, it is freed.

The relay file consists of a set of protocol specific parameters followed by a directory that maps communications identifiers to protocol specific parameters.

Relay File Samples

The following is a sample of a relay configuration file for XCF

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <xcfparms groupname='S6BOSB'/>
   <directory>
      <node name="PRODZDOB">
         <xcf/>
      </node>
      <node name="PRODZNEE">
         <xcf/>
      </node>
   </directory>
</relay>
```

The element and attribute names in the relay file are case sensitive.

The following is a sample of merged relay configuration file for TCP/IP and XCF:

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <tcpipparms tcbnum="3" maxtcbsockets="50" />
   <xcfparms groupname='S6BOSB'/>
   <directory>
      <node name="PRODZDOB">
         <xcf/>
         <tcpip host="zos1.mydomain.com" service="emprec" />
      </node>
      <node name="PRODSDOB">
         <tcpip host="solaris5.mydomain.com" port="26360" />
      </node>
      <node name="PRODZEE">
         <xcf/>
         <tcpip host="zos1.mydomain.com" port="22636" />
      </node>
      <node name="TESTDOB">
         <tcpip host="168.192.0.101" port="26362"/>
      </node>
```

```
   </directory>
</relay>
```

Verifying Relay File Syntax

The S6BRLYVA program allows the administrator of TIBCO Object Service Broker to verify that the syntax of a relay file is correct before starting a TIBCO Object Service Brokercomponent that uses the file. The utility will parse the relay file allocated to the S6BRELAY DD name and will display any error messages in the SYSPRINT DD name.

```
//MYUSERGN JOB (0),'VERIFY PARMS',MSGCLASS=A,NOTIFY=MYUSER,TIME=10
//VALIDATE EXEC PGM=S6BRLYVA,REGION=0M
//STEPLIB  DD   DISP=SHR,DSN=$HLQNONV$.$INSTVER$.AUTH
//S6BRELAY DD   DISP=SHR,DSN=$HLQNONV$.$SLQ$.RELAYCFG
//SYSPRINT DD   SYSOUT=*
```

Refer to the RELAYVAL member in the JCL data set for sample JCL to run the utility.

Appendix C DASD Configuration

This appendix describes how to configure DASD for TIBCO Object Service Broker.

Topics

Plan the DASD Configuration

Device Types

All DASD space requirements are based on single-density 3390-type DASD devices. The TIBCO Object Service Broker data sets are placed on devices that fall into two physical groups: non-VSAM data sets and VSAM data sets, over a range of volumes represented by symbolic variables in *OSEMOD* (*$VOLUM01$-$VOLUM15$*).

Organization of Data Sets

The DASD planning figures by symbolic volume are organized sequentially into two groups:

- Installation data sets ($INSTVOL$)
- TIBCO Object Service Broker system data sets ($OSBVOL$)

Performance Recommendations for DASD Layout

Data sets used by TIBCO Object Service Broker online processing are distributed over a number of symbolic volume names to optimize DASD performance. Other performance recommendations are listed below:

- The Data Object Broker consists of six key operational data sets that should be placed on separate DASD for performance and data integrity reasons. They are the MetaStor, redolog, cache, journals, audit log, and contingency log data sets.
- The redolog and cache data sets should be placed on a DASD with low I/O activity. Otherwise these data sets are good candidates for cache DASD controllers.
- Use load balancing channel and device I/O when positioning MetaStor page data sets and journals, since they have high I/O activity.
- The contingency log has a maximum size of 255 tracks.

Adjust Data Set Sizes for TIBCO Object Service Broker Components

Consider decreasing the default data set sizes for some TIBCO Object Service Broker components due to DASD constraints. The following table lists the space allocations for some components and specifies which member in the CNTL data set to modify. For more information on the CNTL members, refer to Modify Data Sets to Accommodate DASD Constraints on page 435.

Component	CNTL Member	Default Number of Cylinders (3390 Device)	Minimum Number of Cylinders (3390 Device)
MetaStor	SPLXSEG0	100 per page data set	75 per page data set
Segment 1	SPLXSEG1 referenced with last name qualifiers *$SEG1NAM$.PAGEx* in the DASD space allocation tables.	25 per page data set	14 per page data set
Audit log segment (default: segment 99)	SPLXSG*nn* referenced with last name qualifiers *$SEG99NAM$.PAGE1* in the DASD space allocation tables.	50 per page data set	—
Journals	SPLXJRN1 and SPLXJRN2	—	25 per journal
Redolog	SPLXREDO (DPLXREDO for duplex redologs).	—	50

Duplexed redologs must be the same size as their primary data sets.

Modify Data Sets to Accommodate DASD Constraints

If required, use the following tables to further adjust data sets for size and for different devices. The tables contain the following information:

- Column one contains the low-level qualifiers of the TIBCO Object Service Broker data set name.
- Column two contains the CNTL member that allocates the data sets during the install process.
- Column three contains the name of the library where you can find the member.
- Column four contains the primary and secondary (3390) space allocation. Data sets allocated in tracks are rounded up to the next cylinder. Totals are

computed (not including extents) for symbolic volumes that have more than one data set allocated.

- Column five contains the symbolic DASD volume serial number where you intend to allocate the data sets.
- Column six designates the volume as VSAM or non-VSAM. If you do not need to make this distinction at your site, this designator can be ignored.

Required Online Data Sets by Symbolic Volume

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
DBDLIB	DBJCL	JCL	1 TRK	*$OSBVOL$*	VSAM
$SEG0NAM$.PAGE1	SPLXSEG0	CNTL	100/0	*$OSBVOL$*	VSAM
$SEG1NAM$.PAGE1	SPLXSEG1	CNTL	25/0	*$OSBVOL$*	
$SEG99NAM$.PAGE1	SPLXSG99	CNTL	50/0	*$OSBVOL$*	VSAM
$SEG0NAM$.PAGE2	SPLXSEG0	CNTL	100/0	*$OSBVOL$*	VSAM
$SEG1NAM$.PAGE2	SPLXSEG1	CNTL	25/0	*$OSBVOL$*	
$SEG0NAM$.PAGE3	SPLXSEG0	CNTL	100/0	*$OSBVOL$*	VSAM
$SEG1NAM$.PAGE3	SPLXSEG1	CNTL	25/0	*$OSBVOL$*	
ARCHLOG [a]	S6A3ALOC	OSB.JOBS	1 TRK	*$OSBVOL$*	non-VSAM
JRNL1	SPLXJRN1	CNTL	100	*$OSBVOL$*	VSAM
JRNL2	SPLXJRN2	CNTL	100	*$OSBVOL$*	VSAM
CACHE1	S6A3ALOC	OSB.JOBS	35/0	*$OSBVOL$*	non-VSAM
CACHE2	S6A3ALOC	OSB.JOBS	35/0	*$OSBVOL$*	
REDOLOG.PENDING [b]	S6A3ALOC	OSB.JOBS	2	*$OSBVOL$*	non-VSAM
REDOLOG	SPLXREDO or DPLXREDO	CNTL	50/0	*$OSBVOL$*	VSAM

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
RESOURCE	S6A6POST	OSB.JOBS	2/2	*$OSBVOL$*	VSAM
DOBDUMP	OSRUN or OSRUNSTC	JCL	0/80	*$OSBVOL$*	non-VSAM
EEDUMP	NATIVEEE	JCL	0/80	*$OSBVOL$*	non-VSAM
	Total:		649		

a. The ARCHLOG data set must reside within a single extent. A block size of half a track is recommended to ensure that you can save the largest group of records expected. For information on estimating size requirements, refer to the OSBTLFAL (Format ARCHLOG) utility in *TIBCO Object Service Broker for z/OS Utilities*. Before first use, the file must be formatted using OSBTLFAL.

b. The REDOLOG.PENDING data set must be allocated with 2 contiguous cylinders.

Required Offline Data Sets by Symbolic Volume

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
JOURNAL.SPINOUT	S6A3ALOC	OSB.JOBS	100/20	—	non-VSAM
JOURNAL.SPINMRG	SPINMRG	JCL	100/20	—	non-VSAM
	Total:		200		

Optional Data Set Allocation by Symbolic Volume

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
REDOLOG.DUPLEX	DPLXREDO	CNTL	100/0	*$VOLUM06*$	VSAM
	Total:		100		

SMP/E Data Sets

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
AC	S6A3ALOC	OSB.JOBS	5/5 TRKS	*$OSBVOL$*	non-VSAM
ACLST	S6A3ALOC	OSB.JOBS	1/1	*$OSBVOL$*	non-VSAM
AH	S6A3ALOC	OSB.JOBS	5/5 TRKS	*$OSBVOL$*	non-VSAM
ALOAD	S6A3ALOC	OSB.JOBS	15/3	*$OSBVOL$*	non-VSAM (PDSE)
AMAC	S6A3ALOC	OSB.JOBS	2/1	*$OSBVOL$*	non-VSAM
ASRCSAMP	S6A3ALOC	OSB.JOBS	5/5	*$OSBVOL$*	non-VSAM
ASAMP	S6A3ALOC	OSB.JOBS	2/1	*$OSBVOL$*	non-VSAM
C	S6A3ALOC	OSB.JOBS	5/5 TRKS	*$OSBVOL$*	non-VSAM
CLST	S6A3ALOC	OSB.JOBS	1/1	*$OSBVOL$*	non-VSAM
GLOBAL.CSI	S6A3ALOC	OSB.JOBS	6/1	*$OSBVOL$*	VSAM
H	S6A3ALOC	OSB.JOBS	5/5 TRKS	*$OSBVOL$*	non-VSAM
LOAD	S6A3ALOC	OSB.JOBS	15/2	*$OSBVOL$*	non-VSAM (PDSE)
LOG	S6A3ALOC	OSB.JOBS	3/1	*$OSBVOL$*	non-VSAM
LOGA	S6A3ALOC	OSB.JOBS	3/1	*$OSBVOL$*	non-VSAM
LTS	S6A3ALOC	OSB.JOBS	5/5	*$OSBVOL$*	non-VSAM (PDSE)
MTS	S6A3ALOC	OSB.JOBS	1/1	*$OSBVOL$*	non-VSAM
PTS	S6A3ALOC	OSB.JOBS	35/50	*$OSBVOL$*	non-VSAM
SAMP	S6A3ALOC	OSB.JOBS	5/5	*$OSBVOL$*	non-VSAM
SCDS	S6A3ALOC	OSB.JOBS	1/1	*$OSBVOL$*	non-VSAM

Data Set Name	Member Name	Library	3390 Default Pri/Sec (CYLS)	Symbolic Volume	Type
SRCSAMP	S6A3ALOC	OSB.JOBS	5/5	*$OSBVOL$*	non-VSAM
STS	S6A3ALOC	OSB.JOBS	1/1	*$OSBVOL$*	non-VSAM
	Total:		106		

Pagestore Capacity

The Pagestore consists of up to 256 segments. Each segment has 1 to 128 data sets. A TDS (Table Data Store) data set can hold 507,904 x 4 KB pages. Given these parameters, the potential capacity of a segment is approximately 496 gigabytes.

See Also *TIBCO Object Service Broker Application Administion* and *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for more information about segments and modifying the size of your Pagestore.

Appendix D Database Definition Parameters

This appendix describes database definition parameters.

Topics

Overview

DBDLIB

TIBCO Object Service Broker stores physical database descriptions and configuration data in a parameter file called DBDLIB. It is a collection of assembler macro calls that you assemble, link edit, and execute to populate the DBDLIB VSAM data set.

By concentrating all physical system information into one control file, TIBCO Object Service Broker can dynamically allocate data sets and work areas. This simplifies JCL and tuning options.

Sample Member DBJCL

The DBJCL member of the JCL data set contains sample definitions. Part of the member is shown below. In this sample, the first DB statement defines segment 0 (the MetaStor) and the second and third DB statements define additional segments (segment 1 and 99) created as part of the product install.

```
DBSET Q1=S6B.AP,Q2=OSB50
DB    TYPE=PAGE,ACBS=3,Q3=SEG00,SEGNAME=DBSEG0
DB    TYPE=PAGE,ACBS=3,Q3=SEG01,SEGNAME=DBSEG1,ID=1
DB    TYPE=PAGE,ACBS=1,Q3=SEG99,SEGNAME=DBSEG99,ID=99,SYSTEM=Y
DB    TYPE=JOURNAL,ACBS=2
DB    TYPE=REDOLOG,DUPLEX=N
DB    TYPE=RESOURCE

DBSET Q1=S6B.AP,Q2=OSB50
DB    TYPE=CACHE
DB    TYPE=PENDING
DB    TYPE=ARCHLOG

DBGEN
```

If you do not specify the ARCHLOG, it is defined for you.

Database Parameters

Modify DBJCL

Modify the following parameters in DBJCL to reflect your site's requirements.

DBSET and DB Macros

These two Macros are used together to define the Data Object Broker data sets and Segment data sets. The DBSET Macro define global parameters and the DB Macro define local parameters that override the same DBSET parameters. Each invocation of the DBSET Macro reestablishes the global parameters.

DBSET Macro

DBSET parameters define the global data set name qualifiers as described in the following table. All z/OS data set naming conventions apply: the maximum data set name length is forty-four characters and no qualifier can exceed eight characters. The total length of all specified qualifiers cannot exceed thirty-five characters. The Q1= to Q6= keywords must be specified without omitting a lower value. For example, if you specify Q4=, you must specify Q1=, Q2=, and Q3=.

NAME=	NAME= is used as the Segment name when TYPE=PAGE specified and used in the data set name generation following Q1= if Q2= not specified. The maximum length is eight and you can specify NAME= or SEGNAME= but not both.
Q1=	First dsname qualifier or string of qualifiers separated by periods. Each qualifier cannot exceed eight characters, periods not included.
Q2=	Second dsname qualifier or string of qualifiers separated by periods. Each qualifier cannot exceed eight characters, periods not included.
Q3=	Third dsname qualifiers if necessary.
Q4=	Fourth dsname qualifiers if necessary.
Q5=	Fifth dsname qualifiers if necessary.
Q6=	Sixth dsname qualifiers if necessary.

SEGNAME=	SEGNAME= is used as the Segment name when TYPE=PAGE specified and used in the data set name generation following Q1= if Q2= not specified. The maximum length is eight and you can specify NAME= or SEGNAME= but not both.

DB Macro

The DB Macro describes individual Data Object Broker data sets and Segments. The parameters specified on the DB Macro override the parameters specified on the DBSET Macro.

ACBS=1	The maximum number of Pagestore data sets in a segment. The default is 1. This parameter is valid for PAGE and JOURNAL data sets. The base segment requires a minimum value of 3. Journals require a minimum value of 2 and can have as many as 255 data sets. The maximum value for PAGE data sets is128.
FREE=	Specifies whether freed pages are to be journaled and sent to the Pagestore. Valid options are: LOGICAL – A freed page is not to be journaled and not to be written back to the Pagestore data set. This is the default. PHYSICAL – A freed page is to be journaled and written back to the Pagestore data set.
ID=	Specifies the numeric ID of the segment being defined. Default is 0 for the base segment. Each segment must have a numeric ID specified, and it must be unique amongst all the segments. In the sample, segment 1 has a value of 1. Valid range is 0-255.
INIT=Y	Specifies if the segment is to be allocated and opened during Data Object Broker initialization (Y or N). Default is Y. Code N to request the segment not be opened during Data Object Broker initialization. If the Data Object Broker is restarted after an abnormal termination, the value of INIT at the time of termination applies.

JOURNAL=Y	Specifies if updates to data within a segment should be journaled by the Data Object Broker (Y or N). This parameter is valid for page data sets only. Default is Y. Journaling by segment can be controlled by operator commands. Refer to Chapter 18, Monitoring and Controlling the TIBCO Object Service Broker Environment, on page 287 for more information.
MODE=	Use this parameter to restrict write access to any segment other than segment 0 by setting MODE=RO. Absence of this parameter sets the segment for full read and write access. You can also explicitly specify full access to the segment by MODE=RW.
NAME=	Second dsname qualifier when defining PAGE data sets specified by the TYPE=PAGE keyword. NAME= replaces Q2= if Q2 is not specified. The maximum length is eight characters. You can specify NAME= or SEGNAME= but not both.
Q1=	First dsname qualifier or string of qualifiers separated by periods. Each qualifier cannot exceed eight characters, periods not included.
Q2=	Second dsname qualifier or string of qualifiers separated by periods. Each qualifier cannot exceed eight characters, periods not included.
Q3=	Third dsname qualifiers if necessary.
Q4=	Fourth dsname qualifiers if necessary.
Q5=	Fifth dsname qualifiers if necessary.
Q6=	Sixth dsname qualifiers if necessary.
SEGNAME=	Name of segment when defining PAGE data sets specified by the TYPE=PAGE keyword. SEGNAME+ replaces Q2= if Q2 is not specified. The maximum length is eight characters. You can specify NAME= or SEGNAME= but not both.

SYSTEM=	This optional parameter, when set as SYSTEM=Y, designates the segment as a required segment. As the Data Object Broker initializes, it checks for required segments and does not come up if any required segment is inaccessible. After Data Object Broker initialization, a required segment cannot be taken offline.
THREADS=	Defines the number of concurrent I/Os supported per page data set within a segment. If this parameter specifies a value greater than the Data Object Broker MAXVSAMC parameter, the MAXVSAMC parameter value is used. Refer to *TIBCO Object Service Broker Parameters* for a description of this parameter.
TYPE=	Identifies the data set group to which the DB statement refers. Valid options: • ARCHLOG – The archive log. • PAGE – The MetaStor or TDS segments. • JOURNAL – The journals. • REDOLOG – The redolog. • CACHE – The caches. • PENDING – The pending redolog (contingency log). • RESOURCE – The Resource Management Facility repository.
WARN=	When space usage of a segment reaches a minimum threshold—percentage full—the system sends an initial warning message and enables subsequent warning messages to appear. Valid Values: 40 to 100, in increments of 5. Default is 80. Related Parameter is WARNRATE. When a segment is 100% full, limited processing continues: • GET, FORALL, DELETE always proceed. • INSERT proceeds only if the new occurrence fits on an existing page. • REPLACE usually proceeds, unless a new page is needed. • As soon as the transaction needs a new page, processing fails.

WARNRATE	When the segment WARN limit is reached, the system issues more warning messages indicating that the segment could be running out of disk space. A warning is sent out whenever space usage reaches a multiple of the WARNRATE percentage full increment. Valid Values: 5 to 20, in increments of 5, Default is 5. Related Parameter is WARN. For example, if WARN=85 and WARNRATE=5, an initial warning message appears when the segment reaches 85% usage of its total capacity. Subsequent messages are sent out at 90% (85% + 5%), 95%(85% + 2 * 5%), and so on. Netview alerts can be used to immediately identify full segments and recommend action to operations staff. For more information, refer to Chapter 15, TIBCO Object Service Broker Netview Support, on page 253.

DBGEN Macro

The DBGEN macro defines the end of the database definitions. When this macro is encountered, the database definition is generated if all parameters are specified correctly.

See Also

TIBCO Object Service Broker Application Administion for more information about creating additional segments.

TIBCO Object Service Broker Shareable Tools for more information about UTCDATE and UTCTIME.

Appendix E Data Transmission for z/OS

This appendix describes two REXX programs – OSTARXMT and OSTARREC – that are used to simplify transmission of data.

Topics

Overview

TIBCO Object Service Broker provides two REXX programs to simplify transmission of data:

- OSTARXMT – packages data sets, along with error detection information, before transmission.
- OSTARREC – checks for errors and reconstructs data sets after transmission.

The features of these programs include:

- In addition to simplifying transmission (because there is only one data set to transmit), these programs ensure that the data is correct, since error detection is done on z/OS after transmission is complete.
- Support for compressed data for improved transmission.
- Ability to process very large data sets as multiple smaller chunks.
- Help information when called with the '?' parameter.

Installation and Implementation

OSTARREC, OSTARXMT and OSTAREDC are provided as part of the base installation.

Installing OSTARXMT and OSTARREC

To install these programs, do the following:

1. If you already have an existing version of OSTARREC on your system, skip to step #5.
2. Upload to your z/OS system the ostarrec.bin file to a data set with LRECL=80 and RECFM=FB, in BINARY format.
3. Copy the OSTARREC file produced to a data set in your SYSPROC or SYSEXEC concatenation.
4. Allocate a data set on z/OS with a last qualifier of XMT, with LRECL=88 and RECFM=FB.
5. Upload the ostarxmt.xmt file in binary format to the data set you created in step #5.
6. In ISPF 3.4, type OSTARREC against the uploaded data set. You are then prompted for data set names, as documented for the TSO receive command, into which the following data sets should be stored:
   - OSTARXMT, the REXX packaging program
   - OSTARREC, the REXX unpackaging program
   - OSTAREDC, an optional load module to improve the performance of OSTARXMT and OSTARREC
7. To get temporary work data sets with correct DCB information, press <Enter>.
8. Move OSTAREDC to a load library.

   This does not need any special characteristics.
9. Move OSTARXMT and OSTARREC to a PDS in the TSO SYSPROC/SYSEXEC concatenation.

10. Edit OSTARXMT and OSTARREC and change the following line:

```
OSTAREDC = "'HUR01.HURON.LOAD(OSTAREDC)'"
```

to

```
OSTAREDC = "'loadlib.containing.ostaredc(OSTAREDC)'"
```

(the name of the library referenced in step #9).

Implementing the Compression Option

In z/OS Release 1.9, the TRSMAIN program was added to the BCP element of z/OS; so you do not need to download or install it. The main program is called AMATERSE, but the alias TRSMAIN is maintained for compatibility with the previously downloadable version.

To implement compression, do the following:

1. Download the IBM TRSMAIN utility as follows:
   - Access the following web site: http://techsupport.services.ibm.com/390/trsmain.html
   - Download TRSMAIN to your PC as TRSMAIN.bin.
   - Upload this file to z/OS in binary format to a data set with LRECL=80 and RECFM=FB.
   - From ISPF 3.4 against this data set, type:

     ```
     "RECEIVE INDA(/)"
     ```

     When prompted, give the name of the load library where you want the TRSMAIN program restored.
2. Edit the OSTARXMT and OSTARREC REXX programs in turn as follows:
   - Issue the command FIND TRSMAIN 1.
   - Change the constant after the equal sign to contain the full data set name of the program. The string must start with a double quote and a single quote, and end with a single quote and a double quote, because the double quotes delimit the string and the single quotes tell TSO that the data set name is fully qualified. For example:

     ```
     TRSMAIN = "'CSUPP.SHARED.LOAD(TRSMAIN)'"
     ```

     becomes

```
TRSMAIN = "'your.load.library(TRSMAIN)'"
```

(the name of the library referenced in step #1).

Setting up Data Sets for Transmission

To send fixes, objects, and so on, do the following:

1. Use OSTARXMT, on the source z/OS system, to package multiple data sets into one sequential XM1 or XM2 data set.

   If the resultant data set is larger than the CHUNKSIZE value specified, the XM1 and XM2 data sets are divided into multiple smaller data sets with suffixes of XMxFnn. For details on the difference between the XM1 and XM2 formats, see Data Set Formats, page 454.

2. Make the XM1 or XM2 data set available to TIBCO Support by FTPing the data set to ftp.objectstar.com for download via the Internet.

   To request instructions for sending diagnostics on tape or CD, see How to Contact TIBCO Support, page xxi.

3. Load the XM1 file, in binary format, to a sequential z/OS data set with LRECL=1024 and RECFM=FB.

   Keep the original XM1 or XM2 suffix to tell OSTARREC how to process it.

4. Use OSTARREC, on the target z/OS system, to rebuild the original data sets.

All transfers must be in binary format.

Operational Details

OSTARXMT

The OSTARXMT program operates as follows:

1. Prompts for an output data set name.

   The data set must not already exist.

2. Prompts for input data set names.
3. For each input data set, it does the following:
   - Copies the data set to a sequential work data set, with LRECL=80 and RECFM=FB, using the TSO TRANSMIT command.
   - Appends the work data set to the output XMT data set, with LRECL=88 and RECFM=FB, adding an error detection checksum generated by XORing the data both vertically and horizontally.
4. Compresses the XMT file into an XM1 file and then deletes the XMT file.

   The new file name is the same as the XMT file except the last qualifier is XM1.

Data Set Formats

If a single large sequential data set is used as input (for example, an SVC dump), the XM1 file format is not used. Instead, an XM2 format data set is created. The difference between the two formats is, for the XM2 format, TSO TRANSMIT and Error Detection Code are bypassed and only data compression is performed. Creating an XM2 data set takes a fraction of the time it takes to create an XM1 data set and an XM2 data set is slightly smaller because error detection information is not added. The output data set can then be transmitted, as a single data set, by any of the available methods.

File Format Summary:

Suffix	LRECL	BLKSIZE	Description
.XMT	88	n*88	Ensures data integrity by error checking.
.XM1	1024	n*1024	Compressed version of .XMT files. Ensures data integrity by error checking.
.XM2	1024	n*1024	Compressed version of original file. No error checking.

Suffix	LRECL	BLKSIZE	Description
.XM1Fn	1024	n*1024	Compressed version of original file but composed of “chunks.” OSTARREC will combine series of files XM1F1,2,3..n into one final uncompressed file.

Keep suffix on files when you FTP them to TIBCO Support so that OSTARREC will be able to automatically use the correct algorithm to unwrap the file. Always FTP with the binary option.

CHUNKSIZE Option

If the data set is excessively large and you have a slow Internet connection, you can modify the OSTARXMT source to set the CHUNKSIZE value. This divides the XM1 or XM2 data set produced into multiple XM1Fnn or XM2Fnn data sets to facilitate FTP transmission.

Bypassing Prompts

You can bypass the prompting for data set names using the following methods:

- Provide the name of a control data set as a parameter to the program. The format of this data set is as follows:
  - Record 1 contains the words OSTARXMT CONTROL.
  - Record 2 contains the output data set name in any valid TSO format.
  - Subsequent records contain input data set names, one per record.
- Provide a single input data set name. For example, you can type OSTARXMT beside a data set in ISPF 3.4. The output data set has the same data set name except that the last qualifier is changed to XM1.

OSTARREC

The OSTARREC program accepts the input data set name. For example, you can type OSTARREC against a data set in any of the formats in ISPF 3.4.

Error Messages

The OSTARXMT and OSTARREC programs may produce error messages. Other than typing or usage errors, the most likely error is that the ALLOCATE commands used to create work and output data sets do not conform to specific site standards, or do not allocate big enough data sets for some large sets of objects. In this case, find the message listed in this section and modify the appropriate ALLOCATE command in the REXX program. Inform TIBCO Support via a z/OS service request of any such problem.

OSTARXMT Messages

Message	Description
Compression not available.	OSTARXMT cannot find the TRSMAIN program. Refer to Implementing the Compression Option, page 452.
Dataset % already exists.	OSTARXMT wants to create a data set but a data set with that name already exists. Rename or delete the existing data set and try again.
Dataset not found: %. Rejected.	OSTARXMT cannot find the data set. Check that the name was typed correctly using normal TSO naming standards. (If the name is not in single quotes, it is prefixed with your user ID.) If the name is correct, check that the data set is not migrated. If you had a typing error, correct it and retry.
Error allocating dataset %.	OSTARXMT is unable to allocate the specified data set.
Error reallocating XMT file %.	The data set could not be found or created. Review the additional TSO messages.
Error writing to %.	Either the output data set filled up, the volume is full, or you do not have write access to the data set. If the output data set has gone to 16 extents, increase the size allocated for SYSUT1.
Failure adding EDC info. RC=%.	The OSTAREDC program failed. Review the additional TSO messages. The return code is set by the OSTAREDC program.
Invalid space units for file %.	The DCB information of the data set is not supported by OSTARXMT. Review the data set's DCB information. If you think it is valid, contact TIBCO Support.

Message	Description
Only some of the datasets were processed. % is valid.	This is a warning. The data sets that were specified correctly are in the backup.
OSTAREDC program not available.	The procedure was unable to find the OSTAREDC program. Refer to steps # and #9 of “Installation of OSTARXMT and OSTARREC” on page 1.
Read error on temp file.	This should not occur. If it does, record the preceding TSO error messages and contact TIBCO Support.
Unable to allocate file % with DISP=NEW.	The data set may already exist. Otherwise, see other TSO messages and modify the allocation for SYSUT1 to conform to local standards.
Unable to copy % to workfile.	The TSO TRANSMIT command failed to write the input data set to the work data set. If this is a large data set, try increasing the size of the work data set allocated via SYSUT2. Otherwise, check preceding TSO errors.
Workfile 'SYSUT2' allocation failure.	The procedure was unable to allocate a temporary work data set. See other TSO messages and modify the allocation for SYSUT2 to conform to local standards.

OSTARREC Messages

Message	Description
Dataset % already exists.	OSTARREC wants to create a data set but a data set with the specified name already exists. Rename or delete the existing data set and try again.
DSN % suffix % is not XMT, XM1, or XM2.	For OSTARREC to know how to uncompress the file, it needs to know the type of file is supplied. Match the file suffix to the correct type of file.
Error allocating work dataset.	OSTARREC is unable to allocate a temporary work data set. Modify the allocation for SYSUT2 to conform to local standards.
File level error checking failed.	The end of file record has invalid checksum information (it has been corrupted in transit). Download the file again from the server ensuring that you use binary FTP and that there are no errors. Another cause may be that the LRECL is not 1024.
Incomplete XMT dataset. End of file record not found.	After uncompressing an XM1 or XM2 file, the uncompressed work file did not have a valid end of file record. Download the file again from the server ensuring that you use binary FTP and that there are no errors.

Message	Description
INMR152I RECEIVE FAILED. SENDER'S NODE ID NOT RECOGNIZED.	This message is produced by the TSO RECEIVE command, which is used, under the covers, by OSTARREC. It is produced when the TSO TRANSMIT/RECEIVE customization has not been performed at the site when TSO/E was installed.
Input dataset is %. You need a new version of OSTARREC.	The data set you uploaded requires a newer version of OSTARREC. Contact TIBCO Support for a new version.
Invalid input dsn %. Invalid header.	The input data set does not start with the expected header information. Possible causes are a transmission error or a wrong input data set used.
Invalid LRECL=% for dataset with sfx=%.	The logical record length of the data set is incorrect. XMT files should have an LRECL of 88. For XM1 or XM2 files, the LRECL should be 1024. Correct the data set LRECL.
Read error on input dataset.	Unable to read the input data set. See other TSO error messages for details.
Record level error checking failed.	A record has invalid checksum information (it has been corrupted in transit). Download the file again from the server ensuring that you use binary FTP and that there are no errors.
Record/file level error checking failed.	Data set validation failed. Transmission error. Download the file again from the server ensuring that you use binary FTP and that there are no errors.
TRSMAIN not available. Cannot decompress data.	OSTARREC cannot find the TRSMAIN program needed to decompress the input file. For information on identifying TRSMAIN to the program, refer to Implementing the Compression Option, page 452.
Unable to allocate dataset %.	Allocation of the specified data set failed. Check other error messages. Possible causes are security violations or exclusive shares. Also check that the names used exist or do not exist as required.
Unable to find dataset %. Respecify.	The input data set specified does not exist. Check the name and specify an existing data set.
Unable to uncompress %. RC=%.	The TRSMAIN program failed. Review the TRSMAIN messages.

Message	Description
Write error to temp dataset.	The temporary work data set allocated via SYSUT2 is not big enough. Find the allocate statement for SYSUT2 and increase the size.

Appendix F Frequently Asked Questions

This appendix contains five frequently asked questions and their answers.

1. **Q:** Can I specify multiple high-level qualifiers for the installation?

   A: Yes. You have three choices:

   — One qualifier can be for the distribution data sets, which originate from the decompression of the XM1 distribution libraries in the downloaded ZIP file and are in the format `HLQ1` ….

   — A second qualifier can be one for all the data sets allocated during installation.

   — A third qualifier can differentiate non-VSAM qualifiers from VSAM ones. For example, an SMP/E global CSI data set can be `HLQVS2.SMP60.GLOBAL CSI` and a clist library, `HLQ3.INSTALL.CLIST`.

2. **Q: Does the high-level qualifier consist of only a single qualifier?**

   A: No. It can contain multiple qualifiers, for example, `HLQ.QUAL1.QUAL2.SMP60.GLOBAL.CSI`.

3. **Q: How can I differentiate my SMP, installation, and product libraries?**

   A: For differentiation, you can revise three parameters in the properties file:

   – `SPECIFY SECOND LEVEL QUALIFIER FOR SMP/E LIBRARIES: SMP=SMP60`

   – `SPECIFY SECOND LEVEL QUALIFIER FOR INSTALLATION WORK FILES: INSTVER=INSTALL`

   – `SPECIFY SECOND LEVEL QUALIFIER FOR OBJECT SERVICE BROKER SYSTEM FILES: SLQ=OSB`

   You can set the parameter values as you wish. For example, adopting the default values above and specifying a single high-level qualifier allocates the following libraries:

   — `HLQ.SMP60.GLOBAL.CSI` (SMP/E CSI library)

   — `HLQ.INSTALL.CLIST` (installation clist library)

   — `HLQ.OSB.DBDLIB` (product database definition library)

Alternatively, you can set the parameters to an identical value:

— `HLQ.SLQ1.SLQ2.GLOBAL.CSI` (SMP/E CSI library)

— `HLQ.SLQ1.SLQ2.CLIST` (installation clist library)

— `HLQ.SLQ1.SLQ2.DBDLIB` (product database definition library)

4. **Q: With the flexibility of multiple-level qualifications, how can I ensure that the other data sets yet to be allocated do not violate the 44-character restriction?**

   A: Based on the qualifiers you specify in the properties file, the `INSTALL` process, when invoked, determines from the longest possible constructs if a breach exists for the length limitation. If so, the process stops and displays a message that describes the potential violation.

5. **Q: I have installed Object Service Broker 5.2. Can I use the same set of SMP/E libraries for version 6.0?**

   A: No, primarily for maintenance reasons. TIBCO delivers hotfixes as PTFs. You must keep the 5.2 SMP/E environment separate from a newly installed 6.0 version.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z