TIBCO® Object Service Broker

Application Administration

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

TIBCO® Object Service Broker is an application development environment and integration broker that bridges legacy and non-legacy applications and data.This manual provides information required to set up, optimize, administer, and operate your TIBCO Object Service Broker application development environment.

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
TIBCO_HOME *OSB_HOME*	By default, all TIBCO products are installed into a folder referenced in the documentation as *TIBCO_HOME*. On open systems, TIBCO Object Service Broker installs by default into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *OSB_HOME*. The default value of *OSB_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\OSB`. Similarly, all TIBCO Service Gateways on open systems install by default into a directory in *TIBCO_HOME*. For example on Windows systems, the default value is `C:\tibco\OSBgateways\6.0`. On z/OS, no default installation directories exist.
`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*

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
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.
	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`

Table 2 Syntax Typographical Conventions

Convention	Use

`{ }`	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

The TIBCO Object Service Broker Administrator's Workbench

This chapter describes the TIBCO Object Service Broker administrator's workbench.

Topics

Overview

What is the Administrator's Workbench?

The administrator's workbench is the interface to TIBCO Object Service Broker tools, invoked at login, that groups a number of TIBCO Object Service Broker tools and applications together so that an administrator has ready access to these items.

@ADMIN is the default administrator's workbench shipped with the product. You can use the workbench as shipped or you can customize the workbench.

For best viewing results of the workbench and supplied tools, use the model 5 type terminal (132 characters wide). If you use a model 2 type terminal, fields in large tables could display a dot (.). If this occurs, use the Single Occurrence Editor to view the fields.

Who Has Access To This Workbench?

The TIBCO Object Service Broker supplied SYSADMIN user ID, which is a level-7 user ID, is automatically set up to invoke the @ADMIN workbench. A security administrator can set up other user IDs to use the default administrator's workbench by specifying @ADMIN as the login session menu in their user profiles.

Example of the @ADMIN Administrator's Workbench

```
  ADMIN ADMLVL7  TEST: N BROWSE: N  9:44 AM     FRIDAY MAR 16 2007

 1  ER Edit Rule               ==>                          SU MO TU WE TH FR SA
 2  EX Execute Rule            ==>                                       1  2  3
 3  BR Browse Table            ==>                           4  5  6  7  8  9 10
 4  ED Edit Table              ==>                          11 12 13 14 15 16 17
 5                                                          18 19 20 21 22 23 24
 6  OS Object Set              ==>                          25 26 27 28 29 30
 7  DF Define Field            ==>
 8  DT Define Table            ==>
 9  DS Define Screen           ==>
10 DR Define Report           ==>
11 DL Define Library          ==>
12

    COMMAND ==>             __

PFKEYS: 2=LOGS 3=EXIT 12=EXIT
```

Components

The workbench is made up of the following components:

- Common session attributes
- Entry points into the supplied tools
- Messaging mechanisms
- Available PF keys

These components are described in the following sections.

See Also

TIBCO Object Service Broker Managing Security for more information about user profiles and setting the login session menu.

TIBCO Object Service Broker Defining Screens and Menus for information on customizing workbenches.

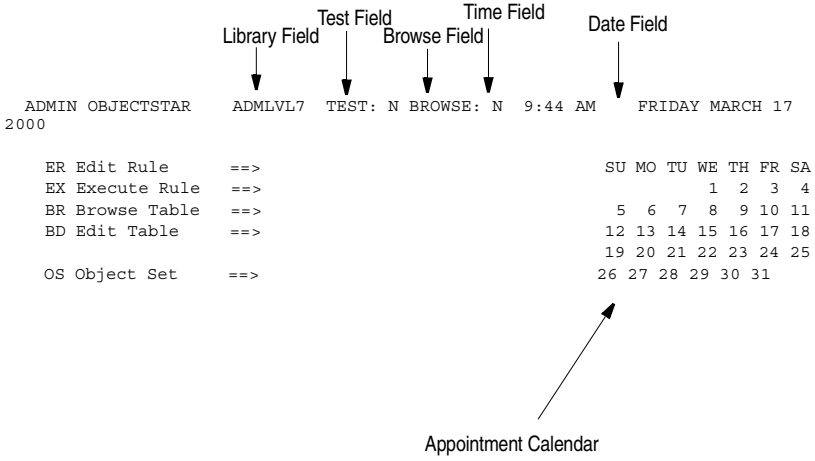
Common Session Attributes

The following session attributes appear at the top of the @ADMIN workbench:

- **Library** field
- **Test** field
- **Browse** field
- **Date** and **Time** fields
- Appointment Calendar

Example

The following figure illustrates the available session attributes:



Library Field

When you first enter TIBCO Object Service Broker, the **Library** field displays the name of the default library specified in your user profile. This is commonly set to your local library, which contains the rules that you created, and the rules from other libraries that you edited and saved, or copied into your library. For more information on library management, refer to *TIBCO Object Service Broker Programming in Rules* and the DEFINE_LIBRARY tool in *TIBCO Object Service Broker Shareable Tools*.

Test Field

The **Test** field applies only for the Execute Rule workbench tool, no other functionality of the workbench is affected. It determines whether a rule, when executed, runs in test mode.

The duration of the **Test** field indicator is transaction bound. When the **Test** field is set to Y, occurrences inserted to TIBCO Object Service Broker tables through a rule are non-persistent even though the rule runs successfully. When the transaction ends or a commit is issued, the updates to tables are discarded. Most external databases are not affected by the **Test** field and changes made to them are persistent.

Browse Field

The **Browse** field applies only for the Execute Rule workbench tool. No other functionality of the workbench is affected. Other tools, like the Menu Definer, give you the option of specifying the mode in which your rule executes. This specification overrides the workbench specification and determines if a rule, when executed, runs in browse mode or not.

When the **Browse** field is set to Y, you cannot make updates to most types of tables through the execution of rules. If you attempt to make an update, the rule fails. Updates are made to EXP tables.

Time and Date Fields

The **Time** and **Date** fields display the current time and date. These reflect the time zone you are in, not the one specified in your base system.

Appointment Calendar

The appointment calendar is a schedule that you can use to record personal messages and reminders. To access the appointment calendar, complete the following steps:

1. Position the cursor on a day number in the calendar or on a field in the reminder area.
2. Press Enter.

   The Appointment Book screen appears.

   If you position the cursor on a day number, the appropriate schedule appears. If you position the cursor on a field in the reminder area, the appointment book for the current day appears.

The Appointment Book Screen

The following example illustrates the Appointment Book screen:

```
        FEBRUARY                  MARCH                    APRIL
   SU MO TU WE TH FR SA     SU MO TU WE TH FR SA     SU MO TU WE TH FR SA
                  1  2  3                     1  2  3      8  9 10 11 12 13 14
   11 12 13 14 15 16 17     11 12 13 14 15 16 17     15 16 17 18 19 20 21
   18 19 20 21 22 23 24     18 19 20 21 22 23 24     22 23 24 25 26 27 28
   25 25 27 28              25 26 27 28 29 30 31     29 30

      APPOINTMENTS:        MONDAY   MAR  17  2000      TO DO:
     8:00                                |
     9:00                                |
    10:00                                |
    11:00                                |
    12:00                                |
     1:00                                |
     2:00                                |
     3:00                                |
     4:00                                |
     5:00                                |
     EVE:                                |
     EVE:                                |
  COMMAND ==>                  Go to month: ___ day#: __ year: _____
```

From the Appointment Book screen, you can type appointments in the appropriate time slots, type items to do in the To Do column, or access different days. To access different Appointment Book screens, you can do either of the following:

- Specify a month, day, and year at the bottom of the screen.
- Use the command line to search for different pages of the appointment book.

Press PF1 for a list of available primary commands for the appointment calendar.

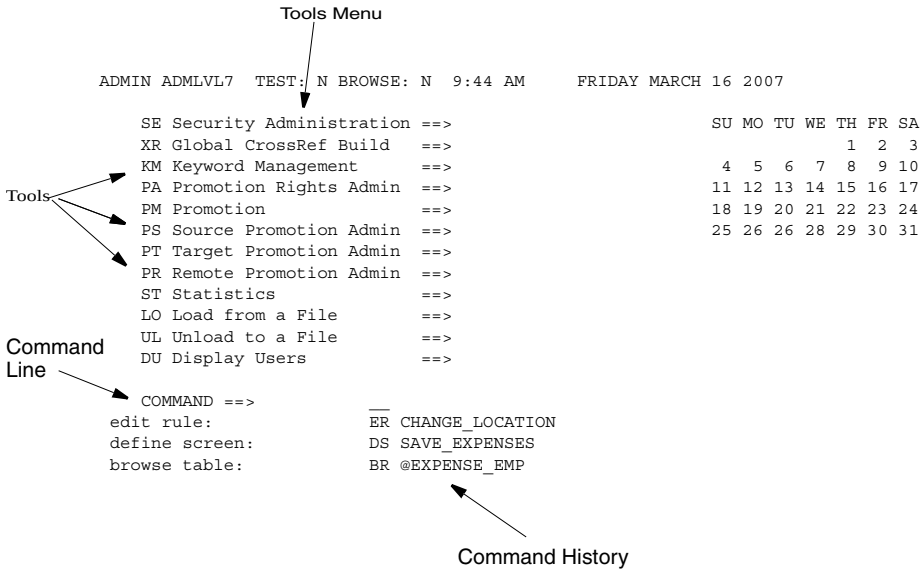
Entry Points into the Supplied Tools

There are three ways to access the tools provided on the workbench:

- Tools menu
- Command line
- Command history area

Example

The following figure illustrates the various entry points into the supplied TIBCO Object Service Broker tools.



Tools Menu

To access a tool from the workbench tool menu, complete the following steps:

1. Place the cursor on the corresponding line for the tool.
2. Type in a TIBCO Object Service Broker object name.

If the name of an object (rule, table, screen, and so on) is not typed, the Object Manager screen is invoked. Refer to *TIBCO Object Service Broker Getting Started* for more information on the Object Manager screen.

3. Type arguments or parameters, if required.

   Rules arguments or table parameters can be supplied with the rule or table name. If the necessary arguments or parameters are not supplied, a prompt screen appears.

4. Press Enter.

Workbench Tools

The following table lists the tools that appear on the @ADMIN workbench and provides a reference to where you can find out more information on a particular tool.

Workbench Tool	Reference
ER Edit Rule	*TIBCO Object Service Broker Programming in Rules*
EX Execute Rule	*TIBCO Object Service Broker Programming in Rules*
BR Browse Table	*TIBCO Object Service Broker Managing Data*
ED Edit Table	*TIBCO Object Service Broker Managing Data*
OS Object Set	*TIBCO Object Service Broker Getting Started*
DF Define Field	FLDMGR tool in the *TIBCO Object Service Broker Shareable Tools*
DT Define Table	*TIBCO Object Service Broker Managing Data* and Chapter 5, Binding, on page 55 to Chapter 7, Mass Updates (z/OS), on page 79.
DS Define Screen	*TIBCO Object Service Broker Defining Screens and Menus* and Chapter 5, Binding, on page 55.
DR Define Report	*TIBCO Object Service Broker Defining Reports*
DL Define Library	*TIBCO Object Service Broker Programming in Rules*

Workbench Tool	Reference
SE Security Administration	*TIBCO Object Service Broker Managing Security*
XR Global CrossRef Build	REFMAKER tool in *TIBCO Object Service Broker Shareable Tools* and *TIBCO Object Service Broker Managing Deployment*
KM Keyword Management	KEYWORDMGR tool in *TIBCO Object Service Broker Shareable Tools*
PM Promotion Rights Admin	*TIBCO Object Service Broker Managing Deployment*
PM Promotion	*TIBCO Object Service Broker Managing Deployment*
PS Source Promotion Admin	*TIBCO Object Service Broker Managing Deployment*
PT Target Promotion Admin	*TIBCO Object Service Broker Managing Deployment*
PR Remote Promotion Admin	*TIBCO Object Service Broker Managing Deployment*
ST Statistics	*TIBCO Object Service Broker for z/OS Monitoring Performance*
LO Load from a File	LOAD tool in the *TIBCO Object Service Broker Shareable Tools*
UL Unload to a File	UNLOAD tool in the *TIBCO Object Service Broker Shareable Tools*
DU Display Users	DISPLAY_USERS tool in the *TIBCO Object Service Broker Shareable Tools*

Command Line

The pair of letters to the left of each tool name is an abbreviation for the tool. You use the command abbreviation in association with the command line.

To access a tool from the command line, complete the following steps:

1. Place the cursor on the command line.
2. Type the appropriate command abbreviation.
3. Type the object name.

   If the name of an object (rule, table, screen, and so on) is not provided, the Object Manager screen is invoked. Refer to *TIBCO Object Service Broker Getting Started* for more information on the Object Manager screen.
4. Type arguments or parameters, if required.

   Rules arguments or table parameters can be supplied with the rule or table name. If the necessary arguments or parameters are not supplied, a prompt screen appears.
5. Press Enter.

Command History Area

The command history area displays a list of previously executed commands. The list displays the last 50 commands issued and is scrollable. To scroll the information in the command history area, move the cursor into the field and press PF7 and PF8 to scroll vertically.

You can use the commands specified in the command history area to issue subsequent ones. To access a tool from the command history area, complete the following steps:

1. Place the cursor on the appropriate command.
2. Modify the command as desired.

   You can change the abbreviation, the object name, argument, or parameter by typing over it before execution.
3. Press Enter.

   The item executes.

Messaging Mechanisms

TIBCO Object Service Broker provides two types of messaging mechanisms from the workbench:

- The broadcast message
- The message log

Broadcast Message

The broadcast message is used to convey a message to all users when they first log in to TIBCO Object Service Broker. It appears at the bottom of the workbench next to the time when the user first logs in to the system. As an administrator you use this field to display a message of interest to users.

The message text for the broadcast message is obtained from the BROADCAST table, which is a shared TIBCO Object Service Broker table. The occurrence with the highest primary key appears on the workbench. If the table is empty, the message `No broadcast available` appears. Users can see the complete table by pressing PF2 before starting another transaction in their current TIBCO Object Service Broker session.

Example

The following screen shows an occurrence from the BROADCAST table:

```
                        --- SINGLE OCCURRENCE EDITOR ---
EDITING TABLE    : BROADCAST
TABLE TYPE       : TDS                                        v
COMMAND ==>
------------------------------------------------------------------------------

NUMBER           : 16841
TEXT             : Promotion scheduled for 2000-03-16 at 19:50:15
                 :




PFKEYS: 1=HELP 2=DOCUMENTATION 3=SAVE 12=CANCEL 13=PRINT 22=DELETE
```

Message Log

Message logs are used to convey information to you when you are processing a rule. TIBCO Object Service Broker maintains two message logs:

- System log
- User log

The System Log

When a rule fails in its execution and the exception is not handled, error messages and debugging information for that transaction are supplied in the system log. Using the information from the system log, you can make corrections to your rule so that it can execute successfully.

The User Log

The user log contains messages generated by rules. It contains the output generated by a transaction (parent) and its descendant (child) transactions. Only output from the most recent transaction is available to display, as output from the previous transaction is cleared when a new transaction begins.

Messages from the system log appear first when PF2 is pressed; pressing PF2 again displays the user log. If there is no message in the system log, the user log appears after the first PF2 command.

See Also *TIBCO Object Service Broker Programming in Rules* for more information on the message log.

PF Keys

PF Keys Available for the Workbench

The following PF keys are available from the workbench. For more information on the available PF keys, place the cursor on the screen outside of a field and press PF1. This invokes screen-level help.

PF Key	Function
PF1	Help.
PF2	Message log.
PF3	Exit.
PF7	Scroll up.
PF8	Scroll down.
PF12	Exit.

Example

The following illustrates the displayed PF keys on the workbench:

```
PFKEYS: 2=LOGS 3=EXIT 12=EXIT
```

See Also *TIBCO Object Service Broker Getting Started* for more information on PF keys.

Customizing Your Workbench

Reasons for Customizing

If you do not want to use the shipped version of the @ADMIN workbench, you can customize it. You can customize your workbench to do the following:

- Exclude various menu options or fields
- Add different menu options or fields
- Change the title area
- Exclude the calendar
- Change the number of commands saved

Requirements to Customize a Workbench

To customize your workbench, you must:

Task	Reference
1. Define a Standard Session Manager menu.	*TIBCO Object Service Broker Defining Screens and Menus*
2. Specify the new menu in your user profile.	*TIBCO Object Service Broker Managing Security*

Chapter 2 Setting Up the Development Environment

This chapter describes how to set up the development environment for TIBCO Object Service Broker.

Topics

Setup Requirements

As an application administrator, you are required to make sure that certain facilities are set up and prepared to manage the application development process. You must consider the following when setting up a TIBCO Object Service Broker development environment:

Security The information that you require to set up TIBCO Object Service Broker security is described in *TIBCO Object Service Broker Managing Security.*

Promotions The information that you require to set up TIBCO Object Service Broker promotions is described in *TIBCO Object Service Broker Managing Deployment*.

Use of global fields A common set of fields, called global fields, can be defined for use in your site's applications. As an administrator you can define global fields and implement how they are used in your development environment. For additional information, refer to Field Dictionary Implementation on page 17.

Implementation of batch processing Batch processing from within TIBCO Object Service Broker makes use of the shared table @SCHEDULEMODEL. This table is initially set up at installation time but you can make changes to its implementation to aid in application administration. For additional information, refer to @SCHEDULEMODEL Implementation on page 19.

Field Dictionary Implementation

As the application administrator, you determine to what extent the global field dictionary, stored in the TIBCO Object Service Broker @GLOBALFIELDS shared table, is to be used when developers at your site are defining tables. You can customize your development environment to:

- Restrict users from using a field that is not included in the global field dictionary
- Issue a warning message if the user includes a field that is not in the global field dictionary
- Allow users to create their own fields without warnings about fields not in the global field dictionary

Global fields are not supported in the graphical Table Definer supplied with the TIBCO Object Service Broker UI. You must use the text-based definer if you are going to use global fields.

How Do You Define Global Fields?

You use the FLDMGR tool to define each global field that is to be included in the @GLOBALFIELDS table. Users, when they are defining tables, screens, or reports can include global fields as part of the definition of the object that they are defining.

See Also

TIBCO Object Service Broker Managing Data, *TIBCO Object Service Broker Defining Reports*, and *TIBCO Object Service Broker Defining Screens and Menus* for more information about defining different objects.

TIBCO Object Service Broker Shareable Tools for more information about the FLDMGR tool.

How is the Use of Global Fields Defined?

Values in the TIBCO Object Service Broker @DT_FLD_CONFIG shared table determine how the global field dictionary is used in your environment. This table, for which you provide the values, determines:

- Types of tables that make use of the global fields option
- Names allowed for fields
- Attributes allowed for fields linked to the global field dictionary

If you leave the @DT_FLD_CONFIG table empty, users can create their own fields or use fields from the global field dictionary at their discretion.

Defining @DT_FLD_CONFIG

The @DT_FLD_CONFIG control table has three fields. Specify the values that your site requires as follows:

TBL_TYPE	Specify the type of table (for example, TDS) for which checks must be made against the global field dictionary. Enter one occurrence for each table type that the checks are made against.
CHK_NAME	Specify whether a user can define a field name that is not linked to a field in the global field dictionary. Valid values are: `R` – a user's field must be linked to a field in the global dictionary or the user is not able to save the table definition. `W` – a user is warned that the field is not linked to a field in the global dictionary and is prompted to confirm before saving the definition of the field. `N or ''` – the field name is not checked against the global field dictionary.
CHK_ATTRI	Specify whether a user can use different attributes (that is, semantic data type, syntax, display length for a screen or report, decimal, or reference) for a field that is linked to the global field dictionary. Valid values are: `R` – a user's field attributes must exactly match the attributes of the field listed in the global field dictionary. `W` – a user is warned that the field attributes do not match those in the global field dictionary and is prompted to confirm before saving the definition of the field. `N or ''` – the field attributes are not checked against the global field dictionary.

@SCHEDULEMODEL Implementation

What is @SCHEDULEMODEL?

The TIBCO Object Service Broker @SCHEDULEMODEL shared table is a parameterized TDS table that contains instances of JCL, Solaris scripts, and batch files. Instances of this table are submitted for processing when users execute rules in batch mode with the use of the SCHEDULE statement. Instances are also used in the TIBCO Object Service Broker Promotion system and by the RULEPRINTER and HLIPREPROCESSOR tools.

@SCHEDULEMODEL supports the use of variable and value substitution; new variables can be defined and subsequently substituted at runtime, and values defined at installation time can be overridden.

Why Allow the Substitution of Variables and Values?

The substitution of variables and values provides a number of advantages to a site:

- New values can be supplied at runtime for variables known to TIBCO Object Service Broker.

  These values take precedence over the Execution Environment parameters set at installation time.

- New variables can be added as the requirements of the site change.
- Users can share instances of the @SCHEDULEMODEL table because information specific to their user IDs can be supplied at runtime.

Implementation Options

At installation time, values used at runtime for some of the instances provided with TIBCO Object Service Broker for the @SCHEDULEMODEL table instances are customized to reflect your site's standards and naming conventions. Values for any or all of these provided instances, or other instances defined at your site, can be supplied by:

- Specifying the values directly
- Referencing the values from a TDS table

Setup Requirements for Substituting Values from a Table

The following items must be prepared to define additional substitution variables and alternate values, by using a TDS table:

- Define and edit a TDS table to hold the names of the variables and user IDs of authorized users.
- Specify {*table.field*} references in the table instances of @SCHEDULEMODEL where you plan to use variables defined in the substitution table.

Define and Edit a TDS Table

To prepare a TDS table, complete the following steps:

1. Define a TDS table with the following characteristics:
   - — It must be non-parameterized.
   - — Define the primary key to hold the values of the TIBCO Object Service Broker user IDs authorized to use instances of the @SCHEDULEMODEL table. The primary key cannot be a composite primary key.
   - — Define a field for each new variable and TIBCO Object Service Broker supplied variable that is to be referenced.

   Any appropriate values can be used for the table name, field names, field syntax, and field lengths.
2. Edit the table so that it contains an occurrence for each TIBCO Object Service Broker user ID that references that table.
3. Specify VIEW_DEFN and READ security access for the table, for each user ID authorized to make use of it.

Sample Table

The following example illustrates a single occurrence for the TIBCO Object Service Broker user ID USR40, from the sample substitution table EXTENDEDPROFILE:

```
                    --- SINGLE OCCURRENCE EDITOR ---
EDITING TABLE    : EXTENDEDPROFILE
TABLE TYPE       : TDS
COMMAND ==>
--------------------------------------------------------------------------------

HURON_USERID     : USR40
ACCTINFO         : ('9061HRNUSR10000',#553)
JOBCLASS         : A
JOBNAME          :
JOBPARM          : SYSAFF=*,TIME=30,LINES=999
MSGCLASS         : Y
NOTIFY           : USR40
REGION           : 4096
MSGLEVEL         : (1,1)




PFKEYS: 1=HELP 2=DOCUMENTATION 3=SAVE 12=CANCEL 13=PRINT 22=DELETE
```

Modify the @SCHEDULEMODEL Instance

To provide a reference between the variable name and its substitution value as stored in the TDS table, complete the following steps:

1. Invoke the Table Editor for the appropriate instance of the @SCHEDULEMODEL table.
2. Insert the table and field name where the value is to be referenced from.

   Use the {*table.field*} format, in the instance of the @SCHEDULEMODEL table. For example, to reference the EXTENDEDPROFILE table for a value for the variable ACCTINFO, put in:

   ```
   ACCTINFO ={EXTENDEDPROFILE.ACCTINFO}
   ```

   Substitution variables in @SCHEDULEMODEL are normally enclosed in delimiters such as braces ({ }). The actual delimiters used at a site are set with the Execution Environment parameters VARLDELIMITER and VARRDELIMITER.

3. Specify VIEW_DEFN and READ security access for the instance for each user ID authorized to use it.

Notes on Editing @SCHEDULEMODEL

- If possible, use a Mod 5 3270 terminal (or its equivalent, if you are emulating a 3270 terminal) to edit the table. If you do not use a Mod 5 terminal, you must edit the table with the Single Occurrence Editor.
- Data entered into the table is case sensitive.

Sample Instance of @SCHEDULEMODEL

The following example illustrates an extract from a sample instance of @SCHEDULEMODEL containing references to the new variables and providing values for new and existing variables.

```
BROWSING TABLE   :  @SCHEDULEMODEL(MVS,DOCEXMPL)
COMMAND ==>
                                                                    SCROLL: P
  NUMBER                                    CARD
_ ------ ------------------------------------------------------------------------
_      10 //{USERID} JOB ({EXTENDEDPROFILE.ACCTINFO}),'TORONTO DOC03',
_      20 // NOTIFY={EXTENDEDPROFILE.NOTIFY},
_      30 // MSGCLASS={EXTENDEDPROFILE.MSGCLASS),
_      50 // REGION={EXTENDEDPROFILE.REGION},
_      60 // MSGLEVEL={EXTENDEDPROFILE.MSGLEVEL}
_      70 /*JOBPARM {EXTENDEDPROFILE.JOBPARM}
_       .
_       .
_       .
_


_
_



PFKEYS: 1=HELP 5=FIND NEXT 9=RECALL 18=EXCLUDE 13=PRINT 3=END 14=EXPAND
```

See Also

TIBCO Object Service Broker Shareable Tools for more information on the RULEPRINTER and HLIPREPROCESSOR tools.

Installing and Operating for your operating environment for information about supplied instances.

TIBCO Object Service Broker Programming in Rules for information about submitting rules in batch.

Chapter 3 Administering a Distributed Environment

This chapter describes how to administer a distributed environment.

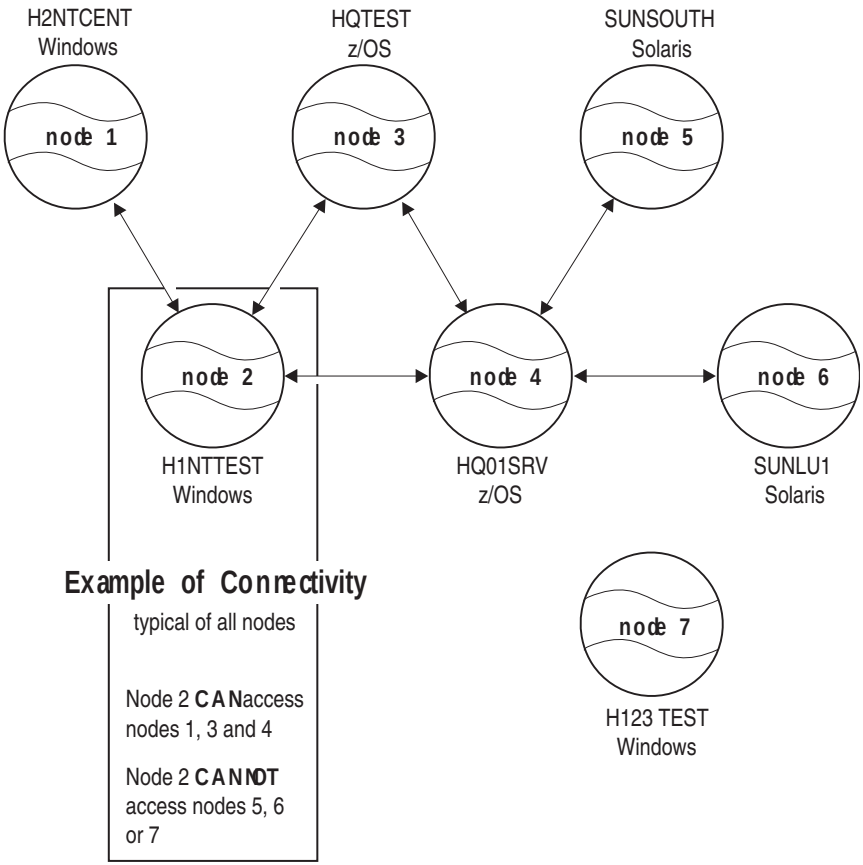
Topics

Implementation of Distributed Data

Supported Inter-connections

If connectivity is established between various TIBCO Object Service Broker systems, users can access data defined on systems other than the one in which they are currently working. When the correct environment is established, TIBCO Object Service Broker ensures that allowable accesses are made to data at the appropriate location.

The inter-connected nodes can be running on any combination of TIBCO Object Service Broker supported operating systems, z/OS, Windows, or Solaris, as shown in this diagram:



Similar connectivity is supported between TIBCO Object Service Broker components: Data Object Broker, Execution Environment, external database servers, and the client interface. For more information about supported connections, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*. For complete details about setting up the communications requirements for this type of distributed configuration, refer to *Installing and Operating* for your operating environment.

Steps Taken to Access Remote Data

The following steps are taken when a request is made from an application for data that is remote to the application:

1. The location of the data is resolved.
2. A check is made to ensure that a peer server is running to access the data at the remote location.
3. Security is checked at the remote location to ensure that the requesting user ID has access to the remote data.

Ensuring Data Integrity

TIBCO Object Service Broker Fail Safe processing ensures that when a transaction ends or a commit is issued, data updates across multiple nodes are handled as if the data were on one node. To make the updates, the transaction must obtain locks for the tables on each node independently.

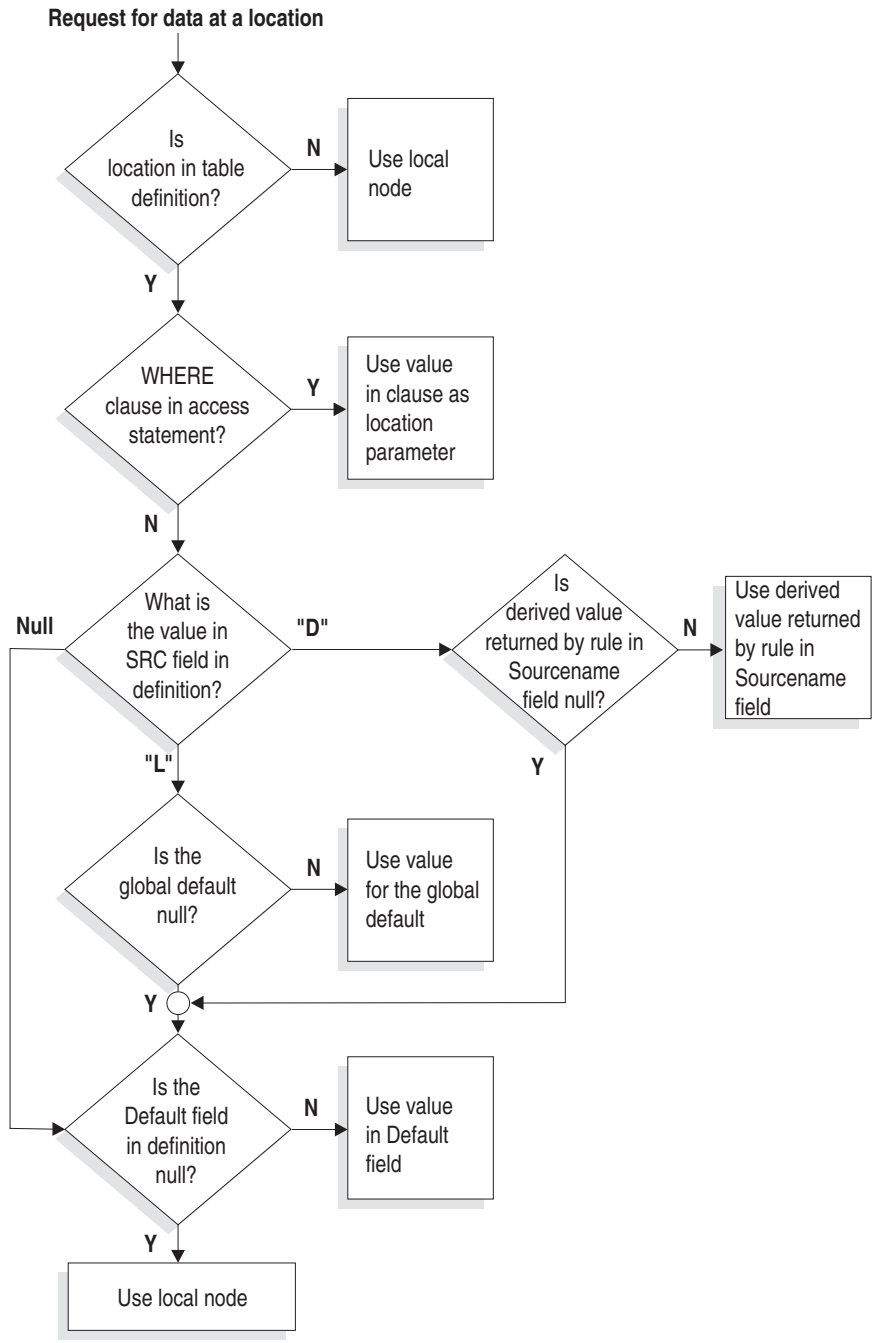
A single transaction can access data in more than one external database, that is, DB2, CA-IDMS, and so on. However, if more than one external database server is used to update data within a single transaction, TIBCO Object Service Broker cannot ensure data integrity.

Locating the Data

For a given table access, the location of the data being retrieved or updated is determined by the application providing a value, implicitly or explicitly, for the location parameter of the table. The value used is the node name for the TIBCO Object Service Broker system where the required data resides. At installation time, a TIBCO Object Service Broker system is assigned a node name using the Data Object Broker parameter NODENAME.

Order of Evaluation to Determine Location

When data is accessed by a rule, the location of the data is determined by a defined order of evaluation. The evaluation is based on a parameter defined as CLASS=L (location) in the table definition. This evaluation continues until either a non-null location value is found or the end of the list is reached. The order of evaluation is given in the following illustration:


Request for data at a location
Is location in table definition?
N
Use local node
Y
WHERE clause in access statement?
Y
Use value in clause as location parameter
N
What is the value in SRC field in definition?
Null
"D"
Is derived value returned by rule in Sourcename field null?
N
Use derived value returned by rule in Sourcename field
Y
"L"
Is the global default null?
N
Use value for the global default
Y
Is the Default field in definition null?
N
Use value in Default field
Y
Use local node

Server Access

If the peer server that is being used to access remote data is busy or if the connection to the peer server fails, one of the following exceptions is raised:

- SERVERBUSY
- SERVERFAIL

SERVERBUSY Exception

The SERVERBUSY exception is raised for the following reasons:

- All the peer servers are in use and therefore a server is not available.
- There are no servers running.
- The value used for the location is invalid.

  Control is passed back to the rule, allowing the rule to try the transaction again.

SERVERFAIL Exception

The SERVERFAIL exception is raised if a transaction is in progress when the connection to the peer server breaks or the peer server fails. Control is passed back to the rule for transaction cleanup.

The exceptions SERVERERROR and SERVERFAIL are also used for access to external databases such as DB2, IMS/DB, or CA-IDMS. For more information about their use by the external database servers, refer to the appropriate *TIBCO Service Gateway* manual.

Binding of Data and Rules

When the remote data and rules are first accessed, the security data and site rules are bound at the remote node in the peer server storage. If changes are made at the remote node that affect security or the site rules (including event and location rules), the peer server storage must be refreshed by shutting down and restarting the server.

For information about starting and stopping a peer server, refer to *Installing and Operating* for your operating environment. For more information about binding, refer to Chapter 5, Binding, on page 55.

See Also For information on configuration requirements between nodes for distributed data processing, refer to *Installing and Operating* for your operating environment.

Setting Up Access to the Data

When the required physical connections are made so that users can access data at a location remote to your node, additional preparation is required. As the administrator, you must make sure:

- Data can be accessed from a default node
- Adequate security is in place at the local and remote nodes
- Adequate data integrity is being maintained

Accessing a Default Store of Data

As described in Order of Evaluation to Determine Location on page 26, the location of data to be used by an application is resolved according to a specific order of evaluation. However, you can set a default location for use in a user's session when a more explicit value is not provided. Two tools are provided with TIBCO Object Service Broker that you can use to manage this default value:

SETREMOTELOC	Sets the value for the default remote location. It takes *value* as its argument.
REMOTELOCATION	Returns the present *value* of the default remote location.

Nodes Available

You can access a list of node names that are available to your local node through the Resource Management facility from the TIBCO Object Service Broker Administration menu.

Changing the Default Location for User Sessions

In some instances it can be necessary for you to change the default remote location for your users. Use SETREMOTELOC and REMOTELOCATION to change this value.

Sample Rule

The following rule changes the default remote location, if it is not already the value required.

```
 RULE EDITOR ===>                                                    SCROLL: P
CHANGE_LOCATION(VALUE);
_
_ ------------------------------------------------------------------------------
_ VALUE = REMOTELOCATION;                                        | Y N
_ ---------------------------------------------------------------+--------------
_ CALL SETREMOTELOC(VALUE);                                      |   1
_ CALL ENDMSG('THE LOCATION IS ' || VALUE);                      | 1 2
_ ------------------------------------------------------------------------------
```

Setting Adequate Security

Security considerations are dependent on the type of security being used: TIBCO Object Service Broker, external, or mixed.

TIBCO Object Service Broker Security

If you specify TIBCO Object Service Broker security using the SECURITY External Environment parameter, security is local to the data and must be defined at the location where it is stored. You must ensure that the following security requirements are met at the local and remote nodes in order for a user to access remote data.

Requirement	Values Required
login ID	Same on both nodes.
Password	Same on both nodes and cannot be null if TIBCO Object Service Broker security is being used to verify login access. If the remote node is on z/OS, the password must be uppercase. [a]
Security groups	Current group of the user ID on the local node must be a group defined on the remote node. The group does not have to be the current group on the remote node and the security permissions for the groups can differ.
Access	Must have VIEW_DEFN access to the table on the local node.

a. For more information on case sensitivity of passwords, refer to *TIBCO Object Service Broker Managing Security.*

External Security

If local user login is authenticated through an external security provider, when accessing a remote node the security requirement for a corresponding user ID and password is no longer applicable.

Mixed Security

In instances where both TIBCO Object Service Broker security and an external security provider are in use, the authentication process checks for TIBCO Object Service Broker security parameters first. If these are not in place, external security parameters are used.

How Do You Ensure Integrity?

In a distributed data environment, if data integrity is to be ensured, you must make sure that the Fail Safe level is set to two on each node that is being accessed.

See Also

Installing and Operating for your operating environment for more information about the Resource Management facility.

TIBCO Object Service Broker Shareable Tools for more information about the SETREMOTELOC and REMOTELOCATION tools.

TIBCO Object Service Broker Managing Security for further information on each of the security types described.

Managing Backup and Recovery for your operating environment for detailed information about Fail Safe levels and processing.

Defining and Storing the Data

Tables with the same names on different nodes do *not* have to be related to each other. A relationship must be explicitly established through the use of a location parameter in the definition of the related tables. When this relationship is established, data can be stored remotely or locally to the definition.

Definitional and Data Requirements

To access data that resides on another node, the following conditions for the table must be met:

- The table containing the data must be defined with the same table name and the same location parameter name on both the local and the remote node.
- If data is to reside on a particular node, the *full* definition for the table must exist on that node.
- If data is not to reside on a particular node, only a minimal definition of that table is required at that node. Optionally, both nodes can have a full definition if this is preferred.
- If data resides on both a local and remote node, the data on *both nodes must be accessible* when the data on the remote node is being accessed.

What is a Minimal Definition?

A minimal definition consists of:

- The table name.
- The location parameter name and attributes.
- The name of the remote node where the full definition is located. This name *must* be supplied through the use of the **Default** field, **Src** field and global default value for the user, or **Src** and **Sourcename** fields.

Binding the Definition

In a production environment, it is advisable that you bind this definition. If this table is not bound, the minimal definition is brought into storage each time the table is accessed. For more information on binding, refer to Chapter 5, Binding, on page 55.

Can a Parameterized Table be Stored Across Nodes?

The location parameter of a table determines the node on which each table instance of a parameterized table can be stored. Two table instances with different values for their location parameter can have the same values for their data parameters. The values for the data parameters do *not* have to be unique across the nodes.

The following conditions apply to storing the data of different table instances across nodes:

- The data in each table instance is specific to the node.
- The full definition must exist on each node that contains data.

Example

The DEPARTMENTS table on NODE1 can have a data parameter value of 10 and occurrences with primary key values of 51, 52, and 58. The DEPARTMENTS table on NODE2 can also have a data parameter value of 10 and occurrences with primary key values of 52, 53, 58, and 59.

See Also *TIBCO Object Service Broker Managing Data* for detailed information about defining tables and inserting data into tables.

Management of Definitions and Data Across Nodes

Some development environments require the developer to maintain consistency with data and definitions that are stored on different nodes. TIBCO Object Service Broker provides a number of tools and facilities to assist in this task. With these, you can:

- Set promotion rights to individual objects at a central location
- Promote objects across nodes
- Manipulate data and definitions across nodes

Setting the Location for Promotion Rights

Using the following shared tables you can centralize where promotion rights are determined for individual objects across the nodes used for development.

@BORROWED_OBJ	This table, parameterized by LOCATION, holds the promotion rights for each object for a node.
@BOROBJLOC	This table can be used to hold the value for LOCATION for @BORROWED_OBJ.

Sample Scenario

The following table illustrates how one or more nodes can use the same source to determine promotion rights.

Node	Value in @BOROBJLOC	LOCATION for @BORROWED_OBJ
NODEA	NODEA	NODEA
NODEB	NODEA	NODEA
NODEC	NODEA	NODEA

Promoting Across Locations

Using the option PR Remote Promotion Admin from the administrator's workbench, you can apply promotions to or from a remote node that is connected to your node. When using Remote Promotion Admin, either the node of the source system *or* the node of the target system must be the same as the node from which you are administering the promotion.

Tools Available to Help Development Across Nodes

The following list describes the tools available to manipulate data and definitions across nodes.

COPY_DATA	Copy data from one table or table instance to another table or table instance.
COPY_DEFN	Copy the definition of an object from a source location to a destination location.
COPYDEFN	Copy the definition of one or more objects. This tool is available on the standard developer workbench.
DELETE_DATA	Delete the data from a table or table instance.
DELETE_DEFN	Delete the definition of an object.
DIFF_DATA	Compare the data of two tables and list the differences.
DIFF_DEFN	Compare the definitions of two objects and list the differences.
DIFFDEFN	Compare the definitions of one or more pairs of objects and list the differences. This tool is available on the standard developer workbench.
PRINT_DATA	Print the data of an object.
PRINT_DEFN	Print the definition of an object.
UNLOAD	Unload table data or definitions of objects.
UNLOAD_DATA	Unload table data.
UNLOAD_DEFN	Unload the definition of objects.

UNLOADLIBRARY	Unload a library of rules.

Transferring Files Across Different Platforms

Because the file format used on z/OS is different from the file format used on Open Systems, if you transfer files using FTP you must reformat them using the z/OS offline batch utility S6BBRFRU. Refer to the S6BBRFRU documentation for information about formatting requirements and FTP procedures.

What Types of Files or Data Set Types Can be Reformatted?

You can use S6BBRFRU to reformat the following types of files or data sets:

- TIBCO Object Service Broker segment archives
- Import/export table files
- Missing BDW Variable Format files
- Workbench unload files
- TIBCO Object Service Broker offline batch utility unload files

See Also

TIBCO Object Service Broker Managing Deployment for more information about setting up different types of promotion environments, that is, restrictive or non-restrictive development and about remote promotions.

TIBCO Object Service Broker Shareable Tools for more information on the shareable tools mentioned.

TIBCO Object Service Broker for z/OS Utilities for detailed information about the use of S6BBRFRU.

Chapter 4 Defining Object Sets

This chapter describes how to define object sets.

Topics

Object Sets

What is an Object Set?

An object set is a collection of TIBCO Object Service Broker objects that are grouped together using the Object Set Definer and can be managed as a single entity, aiding in the application development process. An object set can contain any TIBCO Object Service Broker object, including other object sets.

How Do You Define an Object Set?

The Object Set Definer is used to define object sets by grouping objects in the same application together. You can also specify access permissions for the object set from the Object Set Definer. The Object Set Definer can also be used as a workbench for an application. Instead of defining the components of an application through the different tools, you can use the Object Set Definer as the starting point and define the components from it.

How to View the Message Log

The Object Set Definer also produces a message log. After you define an object, you can access the message log by pressing PF21. The message log usually does not have information in it unless you returned to the Object Definition screen from an object definition tool, for example, the Screen Definer.

Steps to Define an Object Set

To define an object set, complete the following tasks:

	Description	Optional	Refer to page…
Task A	Access the object set definition tool.	N	40
Task B	Specify a valid object type.	N	41
Task C	Specify a value for the Name field.	N	41
Task D	Specify additional object types.	Y	41
Task E	Specify access permissions.	Y	42
Task F	[Optional] View the full object definition.	Y	42

	Description	Optional	Refer to page…
Task G	[Optional] Print the Object Set Definition.	Y	42
Task H	Save the definition.	N	43

The following sections provide detailed instructions for defining object sets.

TIBCO Object Service Broker UI

This chapter describes how to define object sets in TIBCO Object Service Broker using the text-based workbench. You can also perform these tasks using the TIBCO Object Service Broker UI, which provides a graphical environment for TIBCO Object Service Broker development. For details, see the TIBCO Object Service Broker UI online help.

Defining Object Sets

Task A Access the object set definition tool

To access the Object Set Definer tool, use one of the following methods. The *objsetname* is the name of the new or existing object set you are defining.

- Type `define_objectset (objsetname)` at the EX execute rule option on the workbench and press Enter.
- Type an *objsetname* at the OS object set option on the workbench and press Enter.
- Type `os objsetname` at the command prompt and press Enter.

The DEFINE OBJECTSET screen appears.

Example

The following example illustrates the DEFINE OBJECTSET screen for the object set EMPLOYEEINFO:

```
              DEFINE OBJECTSET: EMPLOYEEINFO                  UNIT: USR40
COMMAND ==>                                                   Scroll: P

                        OBJECT TYPE:

      Name              Type          Unit      Author     Created     Modifier
---------------- ----------------  --------  --------   ----------   --------
DELETE_EMPLOYEE  RULE
NEW_EMPLOYEE     RULE
ADD_EMPLOYEE_TIT SCREEN
DEL_EMPLOYEE_TIT SCREEN
DELETE_EMPLOYEE  SCREEN            DOCEXMPL  USR40      1999-07-08   USR40
EMPLOYEE_DATA    SCREEN
EMPLOYEE_INFO    SCREEN
EMPLOYEE_SCR     SCREEN            DOCEXMPL  USR40      1999-03-22   USR40
FCNKEY_SPECS     SCREEN
NEW_EMPLOYEE     SCREEN            DOCEXMPL  USR40      1999-03-06   USR40
EMPLOYEES        TABLE             DOCEXMPL  USR40      1999-11-20   USR40



PFKEYS: 3=SAVE 12=CANCEL 22=DEL 4=SAVE & SEC 5=SELECT OBJS 9=DEFINE 21=MSGLOG
```

Task B Specify a valid object type

In the **OBJECT TYPE** field, specify a valid object type by using one of the following methods:

- Type in a value directly and press Enter.
- Position your cursor on the **OBJECT TYPE** field and press PF1. This displays a list of values from which you can select a value, by typing the **s** line command beside it, and press PF3 to return to the Object Set Definition screen. Press Enter when the value appears in the **Object Type** field.
- Press PF5 without specifying an object type. The Object Selection screen appears and you can select from a list of all types of objects. Refer to Selecting Objects for Object Set Definitions on page 44 for more information.

Task C Specify a value for the Name field

To specify a value for the **Name** field, use one of the following methods:

- In the **Name** field, type the name of an existing object of the type specified in the **OBJECT TYPE** field and press Enter.
- In the **Name** field, type the name of a new object of the type specified in the **OBJECT TYPE** field.

  You can use one of the following options to define the new object:

PF3	Save the object set definition and return to the workbench. Access the appropriate definer for the object and define the object.
PF9	Invoke the Definer for the object type specified. Define the object as required. When you press PF3 or PF12 from the Definer screen, you are returned to the Object Set Definer. Note: not all object type definers can be invoked in this way.

- Press PF5 to display a screen that you can use to select the objects that you require. Refer to Selecting Objects for Object Set Definitions on page 44 for more information.

If the object is a table, refer to Using Tables in Object Set Definitions on page 48 for more information on adding tables to the object set definition.

Task D Specify additional object types

If you want to specify additional object types, repeat those steps. A sub-screen appears for each object type that you are defining.

Task E Specify access permissions

If you want to specify access permissions to the objects, complete these steps:

1. Press PF4 from the Object Definition screen.

   This saves the definition of the object set and invokes the Manage Permissions screen for TIBCO Object Service Broker security.

2. Use the primary command FETCH to retrieve any or all of the securable objects within an object set.

   For more information on the **FETCH** command and access permissions, refer to *TIBCO Object Service Broker Managing Security*.

Task F [Optional] View the full object definition

To view the full definition of your object set, leave the **Object Type** field on the Object Definition screen blank and press Enter.

Task G [Optional] Print the Object Set Definition

The following example illustrates the screen used for printing object set definitions:

```
              DEFINE OBJECTSET: EMPLOYEEINFO
COMMAND ==>

                         ENTER ARGUMENTS FOR PRINTING:


                         LIBRARY       ==>

                         ENVIRONMENT   ==>

                         LOCATION      ==>

                         PARENT ONLY   ==> Y



PFKEYS: ENTER=PRINT 12=CANCEL
```

Printing the Object Set Definition

If you want to print the object set definition, complete the following steps:

1. Press PF13 (Shift+PF1) from the Object Set Definition screen.

   The Print screen appears.

2. Indicate the name of the rules library.

   Only provide a library name if the object set contains rules.

3. Specify the name of the presentation environment.

   Only specify a presentation environment if the object set contains screens or objects that contain screens (for example, menus). The default is 3270.

4. Specify the node name.

   Only provide a node name in the **LOCATION** field if the objects are located on a node other than your home node. If this field is left blank, it defaults to your home node.

5. Indicate if all the objects should be printed.

   If PARENT ONLY is Y, only the definition of the object set (parent) is printed. If it is N, the object set and all the objects it contains (children) have their definitions printed.

Task H Save the definition

Press PF3 from the Object Definition screen to save your definition and exit to the workbench.

Selecting Objects for Object Set Definitions

To select objects for object set definitions or modify a series of objects that are selected by default, complete the following tasks:

	Description	Optional	Refer to page…
Task A	Invoke the Object Selection screen.	N	44
Task B	Narrow the selection scope.	N	45
Task C	Select objects.	N	47

Task A Invoke the Object Selection screen

To invoke the Object Selection screen, complete the following steps:

1. From the TIBCO Object Service Broker main menu, use OS to access the Object Set Definition screen.
2. From the Object Set Definition screen, press PF5.

   The following Object Selection screen appears. The screen displays a series of objects selected by default based on the values in the **UNIT** and the **TYPE**

fields provided in the Object Set Definition screen. The **TYPE** field could be blank if you pressed PF5 before specifying an object type.

```
                        O b j e c t   S e l e c t i o n            Scroll: P
COMMAND ==>
Location:                                                      Select All: N
Library (for RULES):                                         Deselect All: N
Presentation Environment:                                   List Children: N
                                                          Show selection specs: Y
========================== Selection Specification ============================
            Attr   Op       Value
           ------ ---- ----------------
             NAME ____ ________________  AND        unspecified
             TYPE  =   SCREEN__________  AND      attributes are
             UNIT =___ SAMPLAPP         AND        ignored
           AUTHOR ____ _________
================================================================================
         Name              Type         Library      Environment      Unit
   ----------------  ----------------  --------  ----------------  --------
_  @EXPENSE_ITEMS    SCREEN                                       SAMPLAPP
_  @EXPENSE_SELECT   SCREEN                                       SAMPLAPP



PFKEYS: ENTER=UPDATE 3=SAVE SELECTION 12=CANCEL
```

3. Modify the listing as required.

   You can modify the listing by excluding fields that are not required for searching through the list of objects. Using this function helps you by narrowing your object selection in long lists.

You could encounter the Object Selection screen while using other TIBCO Object Service Broker functions, such as the UNLOAD tool and the MANAGE_RIGHTS tool.

Task B Narrow the selection scope

To narrow the selection scope, you can use either of these parts of the screen:

- Specify values in the header portion
- Specify selection specifications

Specifying Values in the Header Portion

You can use the **Location**, **Library**, and **Presentation Environment** fields in the header portion of the screen to narrow the selection scope for objects.

Specifying a Location

In the **Location** field, you can specify a node where the selection criteria are applied. If you do not specify a value in this field, your home node is used.

Specifying a Library

If your selection list contains rules or if the **TYPE** field is empty, you can specify the name of the rules library to be searched. If the **TYPE** field is empty, a library should be specified to ensure that all object types are included in the list. Press PF1 to display a list of valid values from which you can select.

Specifying a Presentation Environment

If your object set contains screens, or objects that contain screens, you can specify the presentation environment in which to search for objects. Press PF1 to display a list of valid values from which you can select.

Listing Child Objects

If some objects have child (dependent) objects associated with them, you can specify that you want to list all the child objects that compose the parent objects.

Specifying Selection Specifications

The middle section of the screen can be used to select items to be included in the definition or to narrow the selection list. You can use more than one type of selection criteria for each object type and you can specify multiple object types within one session. For a list of valid values for each of these fields, position your cursor on the field and press PF1.

Do the following when specifying selection criteria:

When specifying	In the Op Field specify	In the Value Field specify
Name	The logical operator to be used.	The name of the object.
Type	N/A.	The name of the object type.
Unit	The logical operator to be used.	The name of the unit associated with the object.
Author	The logical operator to be used.	The name of the author of the object.

Considerations for Object Type Selection Criteria

Note the following about the object type selection criteria:

- If you do not supply an object type, you must specify a value in at least one of the other selection fields.
- If you only specify an object type and no further selection values, a list of the items for the object type defined in your TIBCO Object Service Broker database appears for further selection.

Task C Select objects

There are two ways you can select objects:

- Select objects using the header portion of the screen.
- Select objects from the list of objects at the bottom of the screen.

Select Objects from Header Portion

Using the **Select All** field, you can specify whether all the items displayed, based on selection criteria, should be copied into the Object Definition screen.

Use the **Deselect All** field to un-select items that you selected.

Select Objects from the List of Objects

After you specify the selection criteria and press Enter, the selected items appear in the bottom portion of the screen. You can select the objects displayed in this section by typing an **S** in the **line command** field beside each object.

After you select the objects that you require, press PF3 to save the selections and return to the Object Set Definition screen. The items are listed in the appropriate type sub-screen within the Object Set Definition screen.

Use the **Deselect All** field if you want to deselect items that you selected.

Using Tables in Object Set Definitions

Adding Tables to the Object Set Definition

When adding table objects, you must supply a value for each of the following:

Defn	Y - Include the definition of the table.
	N - Do not include the definition of the table.

Data	Y - Include the data in the table.
	N - Do not include data in the table.

At least one of these two fields must contain a Y.

Specifying Table Instances

If the object is a parameterized table, position the cursor on its name and press PF6 to display one of two screens for selecting table instances to be part of your definition. The screen that appears depends on whether the table has a parameter value table (table of type PRM) associated with it.

Screen for Table with No PRM Table

The Specifying Table Instances screen appears when the table does not have a parameter value (PRM) table associated with it. Use this screen to individually specify all the table instances that are to be part of the object set definition. If no values are specified, the default is all instances of the table.

```
--------------------------------------------------------------------------------
                SPECIFY INSTANCE OF TABLE:  @EXPENSES
--------------------------------------------------------------------------------

Table Parameters:            REGION =

( ALL DATA: Y )









     < specify all parameter values or set ALL DATA = Y for whole table >

PFKEYS: 1=HELP ENTER=CONTINUE 12=CANCEL
```

Screen for Table with a PRM Table

The Parameters for Table screen appears when the table has a parameter value (PRM) table associated with it. Use this screen to select all the table instances that are to be part of the object set definition. If no values are specified, the default is all instances of the table.

```
                  Parameters for table EMPLOYEES                  Scroll: P
COMMAND ==>                                                    Select All: N
Location:                                                     Deselect All: N
                                                        Show selection specs: Y
========================= Selection Specification ============================
      Selection: REGION LIKE '*'
AND               Op                          Value
                  ---- ------------------------------------------------------
          REGION




================================================================================
  Region
  ----------------
_ A
_ CANADA
_ CENTCANADA
_ MEXICO
_ MIDWEST
_ SOUTHWEST
PFKEYS: ENTER=UPDATE 3=SAVE 12=CANCEL
```

See Also *TIBCO Object Service Broker Managing Data* for information on defining parameter value (PRM) tables.

Copying Object Set Definitions

Why Copy Object Set Definitions?

A skeleton application can be contained in an object set and copies of the object set can be made and customized for different departments, clients, and so on. This is beneficial because you can use standardized components of your business applications. This also decreases the amount of work required for components that are used multiple times.

Creating Copies of Object Set Definitions

To create a copy of an existing object set, complete the following steps:

1. Open the existing object set definition that you want to copy.
2. Rename the object set.
3. Press PF3 to save the definition.

   If you copy the definition, the objects included are the same but the security attributes are not copied.

Deleting an Object Set Definition

There are a number of methods you can use to delete an object set definition. Refer to the following table to determine the appropriate method to use:

If the object set…	Use…
Has been promoted to a target system.	Promotion system for that location.
Has not been promoted and is on your local node.	Object Set Definer or DELETE_DEFN.
Has not been promoted and is on a node remote to the node on which you are presently working.	DELETE_DEFN.

Considerations when Deleting an Object Set

If an object set definition has been promoted to another (target) system, you must submit a change request through the Promotion system (of the source system) to extend the deletion to the target system. If you do not issue a change request to delete the definition, the following occurs:

- The object set exists on the target system and no rights are associated with it on the source system.
- If a new object set with the same name is created on the source system, the creator is unable to promote the object set to the target system because an object with the same name already exists there.

Deleting a Definition Using the Object Set Definer

When you are within an existing definition in the Object Set Definer you can delete a definition by doing one of the following:

- Pressing PF22
- Using the **DELETE** command

In either case you are prompted to confirm the deletion.

Deleting a Definition Using DELETE_DEFN

The shareable tool DELETE_DEFN is available to delete existing definitions. Using this tool, if the correct security access is set up, you can delete definitions across TIBCO Object Service Broker nodes or within your local node. You call DELETE_DEFN from within a rule.

Example Rule

The following rule calls the DELETE_DEFN tool to delete the definition of the DEPARTMENTS object set from Node A. Because an object set definition is being deleted, values are not required for the arguments *library* or *environment*:

```
DELETE_OSDEFN;
_
_ ------------------------------------------------------------------------
_ ------------------------------------------------------------------------
_ CALL DELETE_DEFN('OBJECTSET', 'DEPARTMENTS', '', '' , 'NODEA', | 1
_     '');                                                       |
_-----------------------------------------------------------------------
```

See Also *TIBCO Object Service Broker Managing Deployment* for information about change requests.

TIBCO Object Service Broker Shareable Tools for information on the tool.

Chapter 5 Binding

This chapter describes how to bind and unbind tables, screens, and rules.

Topics

Binding Tables, Screens, and Rules

What is Binding?

Binding is the copying of the definition of a table, screen, or rule, or the data of a table into the shared storage of an Execution Environment when the first access is made to it within a session. Subsequent references within the session access this copy.

Advantages

Binding can significantly reduce I/O between the Execution Environment and the Data Object Broker, therefore optimizing access to the database. If binding is not in place, upon each subsequent reference to a table, screen or rule, a new copy is brought back into storage from the data store.

How Long is the Use of a Bound Copy in Effect?

The use of the bound copy is in effect for the life of the session. If a parameterized table is bound, only the table instances that have been referenced are brought into the shared storage, even though the bind is on the whole table.

If required, you can refresh the bound copy during the life of a session. Refer to Refreshing the Bound Copy on page 65 for more information.

Types of Binding

There are two types of binds:

- Definition

  You can bind the definitions of tables and screens. Refer to Definition Binding on page 58 for more information about binding table and screen definitions. Rules are automatically bound when they are promoted into the installation or system library.

- Data

  You can bind the data of TDS tables. If you bind the data, you must also bind the definition of the table. Refer to Data Binding on page 60 for more information.

Maximum Number of Binds

An Execution Environment parameters determine the limits of bound storage: REGIONTABLESIZE.

If a user tries to access a bound table that cannot fit within the allocation for the Execution Environment, the user's application signals a DEFINITIONFAIL error.

Storage Limits for Bound Tables

In addition to bound storage limits in the Execution Environment, each bound table has a storage limit of 26.5 KB. This does not include additional storage required for indexing. The limit represents the sum of sizes of all occurrences retrieved by the Data Object Broker. The size of additional storage is *(number of occurrences) x (keysize + 4)*.

See Also *TIBCO Object Service Broker Parameters* for more information about the REGIONTABLESIZE Execution Environment parameters.

Definition Binding

What Definitions Should You Bind?

In a production environment, it makes sense to bind the definitions of most of, if not all, your production tables and screens. Use the following table to determine if a table or screen is a good candidate for definition binding:

Type of Usage	Advisable to Bind
Accessed by multiple users in a multi-user environment.	Yes
Accessed frequently by a high volume of transactions.	Yes
Definition does not change.	Yes

Considerations for Definition Binding

If updates are made to the definition while the user is using the bound copy of the definition, the changes do not normally take effect until the user's Execution Environment is recycled or the object is rebound. Refer to Refreshing the Bound Copy on page 65 for more information.

Search Path for Bound Rules

Users should set the search path for their rules libraries to I (installation), in their user profiles; otherwise, the users' local libraries are searched before the installation and system library. Searching the local library first increases response time.

Under normal use, updates do not take place on bound definitions, however updates to definitions made with the MOVTAB, SIXBUILD, and SIXDELETE tools take effect on the bound copy of the table in session storage.

When Should You Unbind a Definition?

Unbind the definition of a table or screen if you want to update its definition. Users who are accessing the bound copy of the table or screen at the same time as the table or screen is being unbound normally continue to access the *original* bound version of the definition until their session storage is refreshed.

See Also *TIBCO Object Service Broker Managing Security* for information about user profiles and search paths.

The @PROMUNBINDOBJS and @PROMBINDOBJS tools in TIBCO Object Service Broker *Shareable Tools* for information about unbinding and rebinding a set of bound objects.

Data Binding

What Types of Data Should You Bind?

Ideal candidates for data binding are application control tables and static data lists containing frequently accessed information (for example, a sales tax table). You can only bind the data of TDS tables and if you bind the data you must also bind the definition. Use the following table to determine if a table is a good candidate for data binding:

Type of Data	Advisable to bind
Used for inquiry transactions.	Yes
Small.	Yes
Data is static.	Yes
Data must be updated often.	No

Under normal use, updates are not allowed on bound data; however, updates to data made with the $CLRTAB tool take effect on the bound copy of the table in session storage.

When Should You Unbind Data?

You must unbind a table if you want to update its data. Users who are accessing the bound copy of a table at the same time as the table is being unbound continue to access the *original* bound version until their session storage is refreshed. Refer to Refreshing the Bound Copy on page 65 for more information.

See Also The @PROMUNBINDOBJS and @PROMBINDOBJS tools TIBCO Object Service Broker *Shareable Tools* for information about unbinding and rebinding a set of bound objects.

Procedure to Bind or Unbind a Table

Binding and unbinding a table consists of the following tasks:

1. Display the list of tables, page 61
2. Specify the required option, page 62
3. Specify what to bind, page 62

These tasks are described in the following sections.

Task A Display the list of tables

From the administrator's workbench, position your cursor beside the DT Define Table option and press Enter. This displays the Object Manager screen for the Table Definer, similar to the following screen. Press PF1 for a list of commands that you can use on the screen.

```
                          List of tables to edit
 Command ==>                                                      Scroll P

          NAME         TYPE CREATOR   CREATED     UNIT              DESCRIPTION*
    ---------------- ---- -------- ---------- -------- ----------------------
  _ ABC1             TDS  USR40    2000-04-07 USR40
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR
  _ LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH
  _ DATATEST         TDS  RPSUSR   2000-03-06 RPSUSR
  _ NRGTEST          TDS  USR40    2000-03-03 USR40
  _ USRWEB           TDS  USR00    2000-02-22 USR00
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL
B-Bind C-Clear D-Delete M-Move P-Prt R-Reset Bind S-Sel X-SIX Bld/Del
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
9:10:46 OK
```

Task B Specify the required option

From the Object Manager screen, use one of the following line commands to specify the required option:

B	Bind the table.
R	Unbind the table.

If you are binding the table, press Enter and complete Step 3. If you are unbinding the table, press Enter to invoke the unbind operation.

Task C Specify what to bind

From the displayed screen enter one of the following values in the WHAT_TO_BIND field:

Y	Bind the definition only.
B	Bind both the definition and data.

Press PF3 or Enter to process the command.

Procedure to Bind or Unbind a Screen

Binding and unbinding a screen consists of the following tasks:

1. Display the List of Screens, page 63
2. Specify the required option, page 64

These tasks are described in the following sections.

Task A Display the List of Screens

From the administrator's workbench, position your cursor beside the DS Define Screen option and press Enter. This displays the Object Manager screen for the Screen Definer, similar to the following screen. Press PF1 for a list of commands that you can use on the screen.

```
                          List of defined screens
 Command ==>                                                      Scroll P
                  Select a screen or enter a primary command
          NAME        CREATOR   CREATED     UNIT    MODIFIER  MODIFIED  TYPE
    ---------------- -------- ---------- -------- -------- ---------- ----
   _ DATA_SRCH_SN1    USR40    1999-05-20 USR40    RPSUSR   2000-03-28
   _ BLANK            USR10    1988-04-12 USR      USR10    2000-03-10
   _ TEST_MASK        TEST     2000-03-08 TEST     TEST     2000-03-08
   _ SESSTEST         RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06
   _ SESSTEST1        RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06
   _ DATA_SRCH_SN2    RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06
   _ TEST_EQD         TEST     2000-02-21 D4EQ     TEST     2000-02-23
   _ PR               AKP99    2000-02-21 AKP99    AKP99    2000-02-21
   _ DETS             USR20    2000-02-11 WEB      USR20    2000-02-11
   _ SUM              USR20    2000-02-11 WEB      USR20    2000-02-11
   _ DFS              ABC20    2000-02-08 ABC20    ABC20    2000-02-08
   _ SNAUSTREP        RPSUSR   2000-02-07 RPSUSR   RPSUSR   2000-02-07
   _ YEAR_REPORT      RPSUSR   1999-12-08 RPSUSR   RPSUSR   1999-12-10
   _ UPDATE           RPSUSR   1999-12-06 WEB      RPSUSR   1999-12-07
   _ REPORT2_SN       USR40    1999-11-03 REPORT   RPSUSR   1999-12-06
   _ DT               USR00    1999-12-02 WEB      USR00    1999-12-02
B-Bind D-Delete P-Print R-Reset Bind S-Select
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
9:19:23 OK
```

Task B Specify the required option

From the Object Manager screen, use one of the following line commands to specify the required option and press Enter to process the command:

B	Bind the screen.
R	Unbind the screen.

Refreshing the Bound Copy

How is the Copy Refreshed?

A new copy of a bound object must be brought into the Execution Environment storage to refresh the data or definition of the object. The environment where users are working determines how a new copy of a bound object is brought into storage.

Refreshing the Execution Environment

To refresh the bound copy, the Execution Environment must be shut down and restarted. In a TSO environment a user can log out and log in again to TIBCO Object Service Broker to refresh the copy. In TIBCO Object Service Broker for Open Systems, to reset the bind for *rules* you must shut down and restart the Data Object Broker.

Refreshing an Active Execution Environment

In a rare situation, in a multi-user environment, there is a need to refresh the copy of the bound rule or object while the present Execution Environment session is active. Two routines, $BINDOBJECT and $BINDRULE, are available to a user with level-7 security clearance to refresh the bound copy within a current session. Use the following routines.

$BINDOBJECT

$BINDOBJECT is called as shown:

```
CALL $BINDOBJECT(class, object);
```

where *class* is the name of the object type, that is SCREEN or TABLE, and *object* is the name of the object to be refreshed.

Procedure

Complete the following steps to refresh the copy:

1. Call $BINDOBJECT.

   $BINDOBJECT must be called for each object that is affected.

2. Modify each object as required.

3. Have all users accessing applications affected by these changes exit the applications.
4. Call $BINDOBJECT again.

   This brings the new version of the object into storage where users can access it.

$BINDRULE

$BINDRULE is called as shown:

```
CALL $BINDRULE(library, rulename);
```

where *library* is the name of the installation library and *rulename* is the name of the rule that is being refreshed.

Procedure

Complete the following steps to refresh the copy:

1. Call $BINDRULE.

   $BINDRULE must be called for each rule that is affected.
2. Modify each rule as required.
3. Have all users accessing applications affected by these changes exit the applications.
4. Call $BINDRULE again.

   This resets the library for the rule to the installation library.

See Also *Installing and Operating* for your operating environment for information about stopping and starting an Execution Environment and a Data Object Broker.

Chapter 6 Secondary Indexes

This chapter describes how to use secondary indexes.

Topics

Using Secondary Indexes

What is a Secondary Index?

A secondary index is an index built from secondary key fields. Unlike a primary field, the values in a secondary index field do not have to be unique.

A secondary index is used to access data efficiently. Within TIBCO Object Service Broker you can define and build secondary indexes on selected fields of TDS table types.

Advantages

Secondary indexes can improve the performance of data access by retrieving data based on secondary index values instead of just the primary key value. If you often search on a table for a value or set of values that is not a primary key value, you can improve the search time by defining a secondary index on the field or fields that you regularly use.

Disadvantages

Secondary indexes incur administrative overhead. As described in the following sections, additional processes are required to define, build, and maintain secondary indexes.

How do you Create a Secondary Index?

You can create secondary indexes:

- Interactively, using the Table Definer from the administrator's workbench
- In batch, using the TIBCO Object Service Broker S6BBRSIX (for z/OS) or hrnbrsix (for Open Systems) utilities

Although you can define fields to be secondary index fields in an empty table, the index is not actually created until data is present.

Criteria for Defining and Building a Secondary Index

You can define up to sixteen fields in a TDS table to be secondary index fields. To achieve optimum performance, the secondary key fields should have values that are evenly distributed with low duplication.

The following criteria also apply for building secondary indexes:

- Secondary indexes *cannot* be built on a numeric field that contains null values.
- If the table is parameterized, a secondary index is created for each table instance.
- The first field of a primary key cannot also be a secondary index field.

What Comprises a Secondary Index?

Secondary indexes are composed of a secondary key value, a primary key value, and a pointer to the data page where the corresponding occurrence resides. The KEY attribute in the table definition determines whether a field is a secondary key field. The following uppercase letters are valid secondary key values:

S	The field is a secondary key field.
Q	The field is both a primary and secondary key field.

Data Access

TIBCO Object Service Broker uses secondary indexes in retrieval when it is more efficient than reading the table data directly. An internal optimizer dynamically estimates each index access and chooses the index (or combination of indexes) that requires the least disk I/Os.

This optimization of data retrieval also occurs for the following data accesses:

- Data accesses to subview (SUB) tables and calculation (CLC) tables, where appropriate, use the source table secondary index, if the source table has secondary indexes.
- The tool COUNTOCCURRENCES uses secondary indexes if the selection is based on secondary index fields.

Selection Criteria

The selection criteria for retrieval can use ranges, and the equal to (=) and not equal (¬=) operators on a secondary index field for TDS tables. If multiple secondary index fields are specified with the equality operator, all are used.

Copying Definitions and Data

If you copy a table definition that has secondary indexes defined, the secondary indexes are not copied. You must define them separately.

If you want to copy the occurrences of a table to a target table that has secondary indexes defined, write a rule to perform the copy and commit the data often. The secondary indexes are being built on the target table during the copy, so more updates are being made than during a normal table copy. This requires you to commit your data more frequently.

Clearing Data

If you want to clear a table with secondary indexes, use one of the following tools:

- The online tool $CLRTAB
- The TIBCO Object Service Broker utilities S6BBRCLR (for z/OS) or hrnbrclr (Open Systems)

See Also *Utilities* for your operating environment for more information on the S6BBRSIX/hrnbrsix and S6BBRCLR/hrnbrclr utilities.

TIBCO Object Service Broker Shareable Tools for more information about the COUNTOCCURRENCES and $CLRTAB tools.

TIBCO Object Service Broker Programming in Rules for more information about committing data.

Procedure to Create a Secondary Index

Creating a secondary index consists of the following tasks:

1. Choose the secondary index fields, page 71
2. Define and Build the Secondary Index, page 71
3. Load the table, page 74
4. Edit the FIELDS table, page 75

These tasks are described in the following sections.

Task A Choose the secondary index fields

Up to sixteen fields, totalling a maximum of 127 bytes, can be defined as secondary index fields. Additional secondary index fields can be defined, up to the maximum of sixteen, at a later time.

Good Candidates

Good candidates for secondary index fields are fields in which the retrieved values identify a useful subset of the data. For example, the primary key of the @EMPLOYEES table is the employee number (**EMPNO**) but searches often take place on the last name (**LNAME**) field because it is easier to remember than the employee number. The last names are not necessarily unique as the employee numbers are but they provide a small enough subset of the data to be useful.

Poor Candidates

Numeric fields that can potentially contain null data should not be used as secondary index fields. An index cannot be created on a table that contains null values in numeric fields.

Fields that contain many duplicate values are also poor candidates for secondary index fields. For example, in the @EMPLOYEES table, if many employees live in the same state or province (**STATE_PROV**), a large subset of data could be returned in response to a data access.

Task B Define and Build the Secondary Index

If you run an index creating tool against an empty table, the index is defined, but not built, because there are no values in the table on which to build index entries. The index is built when you insert data into the table.

If you run an index creating tool against a populated table, the index is both defined and built.

You must have change definition rights to the table to define and/or build the secondary index.

Tools Available

Use the following table to determine the tools to use to define/build a secondary index:

Table Type	Is the table empty?	Using an Interactive Tool	Using Rules	Using a Batch Process
TDS	Yes	Table Definer	SIXBUILD	
TDS	No	Table Definer	SIXBUILD	S6BBRSIX/hrnbrsix

Using the Define Table Option

To use the Define Table option, complete the following process:

1. On the administrator's workbench, position the cursor beside the DT Define Table option and press Enter.

   A list of tables similar to the following appears.

```
                          List of defined tables
 Command ==>                                                    Scroll P
           Enter one or more line commands or a primary command
          NAME        TYPE CREATOR    CREATD     UNIT    MODIFIER   MODIFD    BI*
    ---------------- ---- -------- ---------- -------- -------- ---------- ---
  _ ABC1             TDS  USR40    2000-04-07 USR40    USR40    2000-04-07 N
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH   SEARCH   2000-03-28 Y
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06 N
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 Y
  _ DATATEST         TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ NRGTEST          TDS  USR40    2000-03-06 USR40    USR40    2000-03-06 N
  _ USRWEB           TDS  USR00    2000-03-06 USR40    USR40    2000-03-06 N
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL USR50    2000-03-05 N
B-Bind  C-Clear  D-Delete  M-Move  P-Prt  R-Reset Bind  S-Sel  X-SIX Bld/Del
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

2. Type the line command **X** beside the required table and press Enter.

   A list of fields for that table appears. An example of the display for the @EMPLOYEES table follows:

```
              List of fields in table     TABLENAME: @EMPLOYEE2
Command ==>                                                        Scroll P
           Enter one or more line commands or a primary command
         NAME        TYPE SYNTAX LENGTH DECIMAL PRIMARYKEY SECONDARYINDEX
   ---------------- ---- ------ ------ ------- ---------- ---------------
  _ EMPNO            I    P           3       0 Y
  _ LNAME            S    C          22       0            Y
  _ POSITION         S    C          14       0
  _ MGR#             I    P           3       0
  _ DEPTNO           I    B           2       0
  _ SALARY           Q    P           4       2
  _ ADDRESS          S    V          38       0
  _ CITY             S    C          20       0
  _ STATE_PROV       S    C           4       0
  _ ZP_CODE          S    C           7       0
  _ HIREDATE         D    B           4       0


B-Build SIX  D-Delete SIX
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

3. Type the line command **B** next to fields that you are defining as secondary index fields and press Enter.

   A Y appears in the SECONDARYINDEX attribute for the fields, for example **LNAME** in this example screen. If the table has a composite primary key, you can define a secondary index on any of the key fields except the first one.

Using SIXBUILD

To use the SIXBUILD tool, execute it from a rule by specifying:

```
CALL SIXBUILD(tablename, fieldname);
```

where

tablename	Name of the table where the secondary index is to be built
fieldname	Name of the field on which the secondary index is to be built

A call must be made for each field that is to be a secondary index field. If the table contains data, the index is built when you call SIXBUILD. If the table is empty, the index is built when data is loaded into it.

SIXBUILD cannot be used for large tables (tables with more than approximately 3000 data pages); you must use S6BBRSIX or hrnbrsix instead.

The following example defines and builds a secondary index on the field LNAME in the populated @EMPLOYEE2 table:

```
CALL SIXBUILD('@EMPLOYEE2', 'LNAME');
```

Using S6BBRSIX/hrnbrsix

S6BBRSIX and hrnbrsix are batch utilities that you can use to build secondary indexes. S6BBRSIX is the z/OS version and hrnbrsix is the Open Systems version.

S6BBRSIX/ hrnbrsix

Used to build secondary indexes on populated TDS tables. Indexes on tables of more than 3,000 data pages or that exceed site limitations for online building must be built using this utility.

You must also prepare a set of control cards before invoking the secondary index build utilities. These control cards are defined using the SIXBUILD_CARDS tool.

Task C Load the table

When you define an index on an empty TDS table, the index is not built until you put data in the table. After the indexes are defined, the index entries are built on every occurrence subsequently inserted into the table.

You can use the following to load a table with data and consequently build the index:

INSERT statement	A statement that you can use within a rule to load data into a table.
LOAD	An online tool that loads data into TDS tables.
S6BBRTBL/hrnbrtbl	A batch utility that loads data into TDS tables. When using this utility, you must specify the fields that have secondary indexes.

Task D Edit the FIELDS table

If you used S6BBRSIX or hrnbrsix to build the index, you must use the Table Editor to edit the FIELDS table. To edit this table, complete the following steps:

1. Type Fields (tablename) next to the workbench option ED Edit Table.
2. Press Enter.

   tablename is the name of the table on which you created secondary indexes. For example, to mark the secondary indexes for the @EMPLOYEE2 table, type:
   `Fields(@EMPLOYEES)`
3. Type an uppercase S in the KEYTYPE field of the secondary index fields.

   If the field is part of a composite key, put an uppercase Q in the **KEYTYPE** field to show that the field is both a primary and secondary key.

The letters that you add through the FIELDS table appear in the **KEY** field when you look at the definition through the Table Definer.

See Also *TIBCO Object Service Broker Shareable Tools* for more information about SIXBUILD, LOAD and SIXBUILD_CARDS.

Utilities for your platform for specific information on how to use the S6BBRTBL/hrnbrtbl and S6BBRSIX/hrnbrsix utilities.

TIBCO Object Service Broker Programming in Rules for more information about the INSERT statement.

Deleting and Rebuilding a Secondary Index

A secondary index must be deleted before it can be rebuilt. You can explicitly delete a secondary index using the TIBCO Object Service Broker supplied SIXDELETE tool or the Table Definer option from the administrator's workbench.

Why Rebuild a Secondary Index?

A number of reasons can cause you to need to rebuild a secondary index:

- If you are doing massive insertions or deletions to a table, the data pages could split or merge, even though the secondary index pointer is not updated. If it is not updated, the index is flagged as having invalid page pointers and the primary key value is used to search the occurrences instead. Because this causes unnecessary reads of data you should rebuild an index if you did a large number of updates into a table.
- If the secondary index build failed, for example because a numeric field contained null data, you must rebuild the index. A lowercase s or lowercase q in the **KEY** field of the table definition indicates that an index is unusable.
- If you unload the data and definition of a table from a source system and load the definition and data into a target system, the secondary index information is not retained. You must rebuild the secondary index in the target system.

Steps Required

Deleting and rebuilding a secondary index consists of the following tasks:

1. Delete a Secondary Index, page 76
2. Update the data or correct an index build failure, page 78
3. Rebuild the index, page 78

Task A Delete a Secondary Index

You can use the SIXDELETE tool from within a rule to delete secondary indexes for TDS tables. If the table type is TDS, you also have the option of using the Table Definer from the administrator's workbench.

Using SIXDELETE

To use the SIXDELETE tool, execute it from a rule by specifying:

```
CALL SIXDELETE(tablename, fieldname);
```

tablename	Name of the table from where the secondary index is being deleted
fieldname	Name of the field from where the secondary index is being deleted

A call must be made to SIXDELETE for each field that is having the secondary index deleted. The following example deletes a secondary index from the **LNAME** field in the @EMPLOYEE2 table:

```
CALL SIXDELETE('@EMPLOYEE2', 'LNAME');
```

Using the Define Table Option

To use the Define Table option, complete the following steps:

1. On the administrator's workbench, position the cursor beside the DT Define Table option and press Enter. A list of tables similar to the following appears.

```
                          List of defined tables
 Command ==>                                                    Scroll P
           Enter one or more line commands or a primary command
          NAME        TYPE CREATOR    CREATD     UNIT    MODIFIER   MODIFD    BI*
    ---------------- ---- -------- ---------- -------- -------- ---------- ---
  _ ABC1             TDS  USR40    2000-04-07 USR40    USR40    2000-04-07 N
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH   SEARCH   2000-03-28 Y
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06 N
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 Y
  _ DATATEST         TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ NRGTEST          TDS  USR40    2000-03-06 USR40    USR40    2000-03-06 N
  _ USRWEB           TDS  USR00    2000-03-06 USR40    USR40    2000-03-06 N
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL USR50    2000-03-05 N
B-Bind  C-Clear  D-Delete  M-Move  P-Prt  R-Reset Bind  S-Sel  X-SIX Bld/Del
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

2. Type the line command **x** beside the required table and press Enter.
3. A list of fields for that table appears. The following shows an example of the display for the @EMPLOYEE2 table.

This displayed screen uses a Y to indicate which fields are key fields.

```
              List of fields in table     TABLENAME: @EMPLOYEE2
 Command ==>                                                      Scroll P
            Enter one or more line commands or a primary command
          NAME        TYPE SYNTAX LENGTH DECIMAL PRIMARYKEY SECONDARYINDEX
    ---------------- ---- ------ ------ ------- ---------- --------------
  _ EMPNO             I    P           3       0 Y
  _ LNAME             S    C          22       0            Y
  _ POSITION          S    C          14       0
  _ MGR#              I    P           3       0
  _ DEPTNO            I    B           2       0
  _ SALARY            Q    P           4       2
  _ ADDRESS           S    V          38       0
  _ CITY              S    C          20       0
  _ STATE_PROV        S    C           4       0
  _ ZP_CODE           S    C           7       0
  _ HIREDATE          D    B           4       0


B-Build SIX  D-Delete SIX
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

4. Type the line command D next to fields that you are deleting as secondary index fields and press Enter. The secondary key specification for the fields is deleted from the table definition. If the field is part of a composite primary key, the Q displayed in the table definition is changed to P.

Task B Update the data or correct an index build failure

The reasons you are rebuilding the secondary index determine the actions that you take. Use the following table to determine the appropriate action:

Reason for the Rebuild	Action
Large updates made to a table.	Update your data.
Unloaded data and definition from a source system.	Load the definition and data onto the target system.
An index build failure occurred.	Correct the reason for the failure.

Task C Rebuild the index

Rebuild your index as described in Procedure to Create a Secondary Index on page 71.

Chapter 7 Mass Updates (z/OS)

This chapter describes how to perform mass updates in z/OS.

Topics

Mass Updates of TDS Data

You can do mass updates of large TDS tables in TIBCO Object Service Broker using the batch utilities S6BBRTBL/hrnbrtbl (batch load) or S6BBRCLR/hrnbrclr (batch table clear). Within the z/OS environment you can also use the mass update feature within a rule.

What Method to Use?

Use the following table to determine the method to use to do a mass update of data:

If you want to…	Use…	For more information refer to…
Load a table with initial data.	S6BBRTBL/hrnbrtbl.	*Utilities*
Clear a table or table instance of all its data.	S6BBRCLR/hrnbrclr.	*Utilities*
Selectively insert, replace, or delete data in a z/OS environment.	The mass update feature from within a rule.	The following sections.

Advantages of the Mass Update Feature

You can use the mass update feature to:

- Improve performance because it takes a lock at a table level and not at an occurrence level. This decreases message traffic.
- Manipulate, through rules processing, the data prior to updating the table.
- Make global or selective changes to a table or table instance.
- Bypass the existence test of the two-phase commit process.
- Keep the segment where the table is located online, so that journaling can be performed.

Using the Mass Update Feature

What Conditions Apply?

The following conditions apply to making mass updates to data:

- Changes can only be made to the logical table, not a subview of the table.
- The occurrences must have unique keys.
- When the first non-duplicate insert succeeds, insert failures for duplicate key conditions are not recognized until a commit is issued or the end of the transaction is reached, and an unrecoverable error is signalled.
- If the table is parameterized, during the mass update other transactions can update table instances that are not being updated. Other transactions cannot update an un-parameterized table during the mass update.
- The IDgen attribute must be set to F. The IDgen attribute must be re-set to N or Y after the mass update to re-invoke the two-phase commit process.
- Updates take place faster if the data is in the correct sort order, by parameter and primary key.

Modifying the IDgen Attribute

The IDgen attribute, which is set as part of a table definition, is used to determine whether TIBCO Object Service Broker should generate values for the primary key field. This field accepts the following values:

Y	The system generates a unique value for the primary key of each occurrence when data is inserted into a table.
N	Users who insert data into the table must enter a unique value for the primary key of each occurrence.
F	Value is required for the mass update feature.

Methods Available

The following methods are available to you to modify the IDgen attribute:

- Using the Table Definer
- Using rules

If the table definition has been bound, that is, FIX=Y, you cannot dynamically change the IDgen setting unless the table is unbound. Refer to Procedure to Bind or Unbind a Table on page 61 for more information about binding.

Using the Table Definer

To use the Table Definer option, complete the following steps:

1. From the administrator's workbench, type the name of the table beside the DT Define Table option and press Enter.

   A screen similar to the following appears.

```
COMMAND==>                          TABLE DEFINITION

     Table: @EMPLOYEES       Type: TDS   Unit: DOCEXMPL           IDgen: N
    Source:
  Parameter Name  Typ Syn Len Dec Class             '      Event Rule   Typ Acc
  ----------------  -   -  --- --    -              '   ---------------- -   -
_ REGION            I   C   16  0    D              ' _ DELETEMPNO       T   D
_ LOCATION          I   C   16  0    L              ' _ VALIDEMPNO       V   I
     Field Name   Typ Syn Len Dec Key Ord Rqd    Default          Reference
  ---------------- -   -  ---- --  -   -   - ---------------- ----------------
_ EMPNO            I   P     3  0  P
_ LNAME            S   C    22  0  S
_ POSITION         S   C    14  0
_ MGR#             I   P     3  0                                 MANAGER
_ DEPTNO           I   B     2  0
_ SALARY           Q   P     4  2             0.00
_ ADDRESS          S   V    38  0
_ CITY             S   C    20  0
_ STATE_PROV       S   C     4  0
_ ZP_CODE          S   C     7  0
_ HIREDATE         D   B     4  0
_
PFKEYS: 3=END 12=CANCEL 22=DELETE 13=PRINT 14=FIELDS 21=DATA 2=DOC
```

2. Type an F into the IDgen field and press PF3.

Using a Rule

You can use a rule similar to the following to change the IDgen attribute:

```
RULE EDITOR ===>                                                    SCROLL: P
SET_IDGEN(IDGEN_VALUE, TABLENAME);
_
_ ------------------------------------------------------------------------------
_ ------------------------------------------------------------+---------------
_ GET TABLES WHERE NAME = TABLENAME;                          |   1
_ TABLES.IDGEN =IDGEN_VALUE;                                  |   2
_ REPLACE TABLES;                                             |   3
_ ------------------------------------------------------------------------------
```

Sample Rule to Do a Mass Update

The following rule uses:

1. The SET_IDGEN sample rule to set the IDgen attribute of a table to F.
2. The LOAD tool to load data into a table.
3. The SET_IDGEN sample rule to reset the IDgen attribute to N.

```
RULE EDITOR ===>                                                    SCROLL: P
LOAD_TABLE(TABLENAME);
_
_ ------------------------------------------------------------------------------
_ ------------------------------------------------------------+---------------
_ EXECUTE SET_IDGEN('F', TABLENAME);                          |   1
_ CALL LOAD('USR40.TEST.UNLOAD(MANAGER)', 'SCR');             |   2
_ EXECUTE SET_IDGEN('N', TABLENAME);                          |   3
_ ------------------------------------------------------------------------------
```

Chapter 8 Configuring Your Pagestore on z/OS

This chapter describes how to configure the TIBCO Object Service Broker Pagestore in z/OS.

Topics

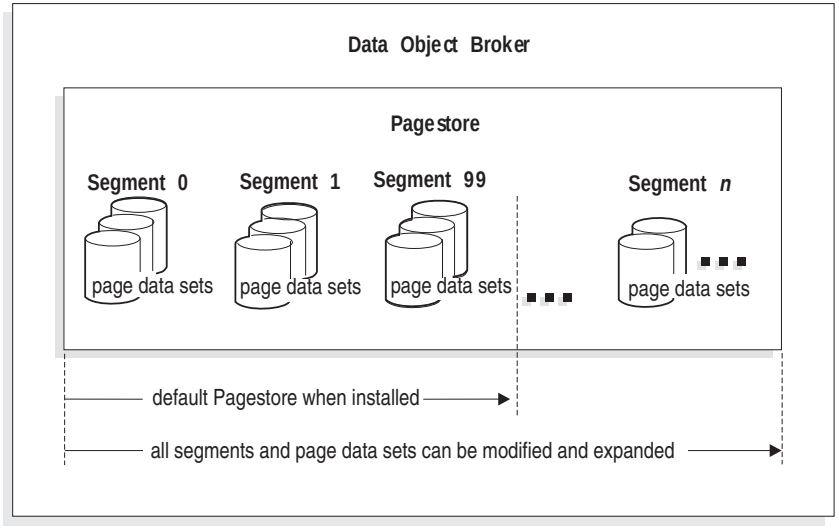
TIBCO Object Service Broker Data Store Architecture

The Data Object Broker accesses data maintained in the TIBCO Object Service Broker data store, called the Pagestore, which is divided into segments. The Pagestore is the repository for all segments associated with a specific Data Object Broker.

With TIBCO Object Service Broker, you can isolate or group your data and store it in separate segments of the Pagestore, according to the use and sensitivity of the data. With your Pagestore divided, you can also take one segment offline, back it up, clear the data, and reload it while keeping all other applications online. As your database grows, your Pagestore can be enlarged or additional segments can be added as needed.

Relationships

The following illustration shows the relationship of the Data Object Broker, the Pagestore, and the segments within the Pagestore.



TIBCO Object Service Broker Pagestore

How is TIBCO Object Service Broker Installed?

When you install TIBCO Object Service Broker, your Pagestore consists of a *minimum* of three segments.

Segment 0 — This is the base segment. It contains the MetaStor, which is made up of definitions, characteristics, access paths, and storage locations of all data and programs.

Segment 0 contains three page data sets when it is distributed.

Segment 1 — This segment is set up to help you separate your user data from the dictionary information in segment 0.

Segment 1 contains three page data sets when it is distributed.

Segment 99 — This segment is used to hold Security audit data.

What is the Logical View of the Data?

The logical view of the data is stored in the form of tables. The data for a table resides completely within a specific segment and can span all the physical data sets comprising the segment. The definition of the table is stored in segment 0, which is known as the MetaStor.

How is the Data Physically Stored?

Data is ultimately stored within page data sets as 4096 byte pages. Each 4 KB page has a header record that identifies:

- The segment to which it belongs
- The page data set within the segment
- The page number (VSAM RBA of the control interval) and type
- Page chaining information
- The data size and number of occurrences
- Date and time of the last update to this page
- Checkpoint/transaction information of the update

The pages are written to the page data sets by the Data Object Broker as part of checkpoint processing.

What Changes Can be Made to the Pagestore?

As the requirements of test or production systems change, some people need to increase or decrease the amount of free space in the Pagestore. Free space can be modified in the Pagestore in three ways:

- You can increase or decrease the size of existing Pagestore data sets.
- You can partition your Pagestore by adding additional segments.
- You can add additional data sets to an existing segment.

You can also restore the Pagestore, or segments of the Pagestore, from backup

See Also *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for more information about restoring the Pagestore from backup, and about moving the Pagestore.

Modifying the Size of a Page Data Set

As your system requirements change, you can increase or decrease the size of an existing page data set. TIBCO Object Service Broker provides utilities to back up your data, allocate new data sets, and restore your data to the new data sets.

This process does not re-organize your data set. Each page data set has the same number of used pages after the restore and each page is restored to its original location in the page data set.

Ensure that the resized page data sets are at least large enough to accommodate the highest page number in the backup file that is to be restored.

Increasing or Decreasing the Size of a Page Data Set

To change the size of a page data set, complete the following steps:

1. Shut down your TIBCO Object Service Broker system or take the segment offline if it is not segment 0.
2. Backup the segment whose data sets are to be modified using S6BTLBPS or S6BTLUPS.
3. Run the pointer checker utility S6BBRPTR against the backup.
4. If problems are reported by S6BBRPTR, fix the errors before proceeding.
5. If necessary, repeat the three previous steps until no problems are reported. Validation of the integrity of data sets can require the running of S6BBRPTR more than once.
6. Delete the old data sets.
7. Redefine the new data sets using IDCAMS.
8. Format the new page data sets using S6BTLFPS.
9. Restore the previous clean backup using S6BTLRPS.
10. Take a full segment backup and run it through S6BBRPTR.
11. Merge the last backup into the continuous backup process, if it is being used. The last full backup replaces the continuous backup master copy.
12. Restart the Data Object Broker or bring the segment back online.

See Also *TIBCO Object Service Broker for z/OS Utilities* and the *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for information about the utilities.

Defining an Additional Segment

Tasks Required to Define Additional Segments

You can define additional segments as your system requirements increase. To define an additional segment, complete the following tasks:

Task A Edit and save the sample JCL

1. Edit the sample JCL S6A5DBDG in the OSB.INSTALL data set.
2. Locate the "DB TYPE=PAGE,ACBS=*n*,…" statement.
3. Replicate this statement.
4. Replace the values.

   Refer to *TIBCO Object Service Broker for z/OS Installing and Operating* for more information about this sample JCL.
5. Save the edited version of the file.

Sample JCL

The following sample JCL defines five segments. The first, segment 0, uses the installation supplied name. The second, segment 99, uses the installation supplied name. The next three page segments are named SALES, EXPENSES, and PURCHASE.

```
//ASM.SYSIN DD *
DBSET Q1=S6B
DB TYPE=PAGE,ACBS=3,NAME=SEGMENT0
DB TYPE=PAGE,ACBS=1,NAME=SEGMNT99,ID=99,SYSTEM=Y
DB TYPE=PAGE,ACBS=3,NAME=SALES,ID=01
DB TYPE=PAGE,ACBS=3,NAME=EXPENSES,ID=02
DB TYPE=PAGE,ACBS=3,NAME=PURCHASE,ID=03
DBSET Q1=S6B,NAME=OSB50
DB TYPE=JOURNAL,ACBS=2
DB TYPE=REDOLOG
DB TYPE=RESOURCE
DB TYPE=CACHE
DB TYPE=PENDING
DBGEN
```

Task B Define new data sets

Use IDCAMS to define the new data sets to be added to the segment. All z/OS naming conventions apply to the naming of the data set. The data set names must be:

```
Q1.NAME.PAGEn
```

Example

Following the sample JCL, the data set names for data in segment 1 would be:

```
S6B.SALES.PAGE1
S6B.SALES.PAGE2
S6B.SALES.PAGE3
```

Task C Format the data sets

After you define the new data sets, you must format them to prepare them for the data:

1. Update the S6A5FRMT member in the OSB.INSTALL data to include new DD lines with calls to S6BTLFPS for the Q1.NAME.PAGEn data sets.

   Ensure that you specify only the *new* data set names in your DD statements. S6BTLFPS fails if you try to initialize data sets that are not empty.

2. Save S6A5FRMT and submit it for execution.

   Upon completion, the new data sets are initialized.

Task D Shut down the TIBCO Object Service Broker system

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs if necessary.
3. Shut down the servers that are running.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

   For details, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Task E Submit the S6A5DBDG for execution

When you submit the S6A5DBDG for execution it executes in the same manner as it did during the initial installation process. As part of the job, other routines are called that assemble and link edit the updated database information. Output is sent to the DBDLIB.

Task F Restart the Data Object Broker

From the system console, issue the command to start the Data Object Broker as shown in the following example:

/F OSTARDOB

For details, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Adding a Page Data Set to an Existing Segment

If a segment is nearing its data capacity, you can use one of two procedures to add one or more page data sets to the segment:

1. Use backup and restore utilities along with segment balance. This is the recommended method especially if the segment has many tables.
2. Use unload and load utilities.

See Also *TIBCO Object Service Broker for z/OS Utilities* and *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for specific information about the utilities mentioned in the following sections.

Procedure 1: Using Backup and Restore Utilities

Complete the following tasks to add a page data set to an existing segment.

Task A Take the segment offline, using the dboffline command

From the system console, issue the following command replacing *nn* with the segment number:

/F OSTARDOB, DBOFFLINE=*nn*

Task B Perform a backup of the segment

1. Edit the JCL for S6BTLBPS (Backup Page Data Sets utility) and change the SEG#=0 parameter to indicate the segment to be backed up.
2. Submit the job to create the backup.

Task C Verify the backup is good

Use S6BBRPTR (Batch Pointer Check utility) to check the backup. If any errors are detected, correct the problems before continuing. Once you have obtained a clean backup, save the backup for the expansion of the segment and recovery, in case it is needed.

1. Edit the JCL for S6BBRPTR and change PARM='SEGMENT=01,HDR' to the segment number that is being processed.
2. In the same JCL, change the //BACKUP DD statement to point to the backup created in Task B.

Task D Increment the number of data sets

1. Edit OSB.INSTALL(S6A5DBDG) to increase the number of data sets by incrementing the ACBS=*n* parameter for the segment.
2. Submit the job OSB.INSTALL(S6A5DBDG) to activate the definition revisions.

Task E Modify your segment allocations

1. Edit the member CNTL(SPLXSEG*n*) or CNTL(SPLXSG*nn*), which describes your segment allocation:
   a. Add a DELETE statement
   b. Add a DEFINE CLUSTER statement for the new page data set being added.
2. Create a job OSB.INSTALL(S6A3ATMP) modelled after OSB.INSTALL(S6A3ALOC) and include only 1 IDCAMS step that contains the new DEFINE CLUSTER statements:

```
OSB.INSTALL(S6A3ATMP)
//*-------------------------------------------------------------------*
//*         Allocate SEGMENT nn new data sets.
//*-------------------------------------------------------------------*
//STEP4     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=$SYSPRT$
//SYSIN     DD DSN *
DEFINE CLUSTER (NAME($HLQVSAM$.$SLQ$.$SEGnnNAME$.PAGEn) UNIQUE -
 VOLUMES($OSBVOL$) NONINDEXED RECSZ(4089,4089) CYLINDERS(20,0) CISZ(4096))
```

3. Submit OSB.INSTALL(S6A3ATMP) to allocate the segment data sets.
4. Using the JCL in OSB.INSTALL(S6A5FRMT) as a model, create a member OSB.INSTALL(S6A5FTMP) to format the segment's data sets using the S6BTLFPS utility. Include a DD statement for each new page data set within the segment:

```
OSB.INSTALL(S6A5FTMP)
     //*-------------------------------------------------------------------*
     //*         Format SEGMENT nn data sets                               *
     //*-------------------------------------------------------------------*
     //FRMTSEGS EXEC PGM=S6BTLFPS
     //STEPLIB  DD DSN=$HLQNONV$.$INSTVER$.AUTH,
     //         DISP=SHR
     //DBDLIB   DD DSN=$HLQVSAM$.$SLQ$.DBDLIB,
     //         DISP=SHR
     //SYSPRINT DD SYSOUT=$SYSPRT$
     //*        INCLUDE ONLY THE NEWLY ALLOCATED PAGE DATASET(S) HERE
     //DD1      DD DISP=SHR,DSN=$HLQVSAM$.$SLQ$.$SEGnnNAM$.PAGEn
     ...
     /*
```

Task F Perform a backup of the expanded segment

Use S6BTLBPS (Backup Page Data Sets utility) to backup the expanded segment.

Task G Verify the expanded data set

1. Use S6BBRPTR (Batch Pointer Check utility) to verify that the expanded data set is good.
2. Correct any detected errors before continuing.

At this point the expanded segment can be used by the Data Object Broker by skipping to Task M, Shutdown the Data Object Broker, on page 97; however, we recommend that S6BBRBAL (Segment Balance utility) be used to balance the data across all the segment's page data sets. This will distribute I/O for new data across all page data sets, yielding better performance.

Task H Distribute the segment data across the page data sets

S6BBRBAL (Segment Balance utility) takes your backup and creates a new backup data set reflecting evenly populated page data sets.

1. Edit the JCL for S6BBRBAL by customizing the PARM settings and data set names for DD statements NEWARC and OLDARC.

Task I Verify the balanced segment

1. Use S6BBRPTR (Batch Pointer Check utility) to verify that the expanded data set is good.
2. Correct any detected errors before continuing.

Task J Reallocate and format the page date sets

1. Edit the OSB.INSTALL(S6A3ATMP) member created in Modify your segment allocations, page 94 and change the IDCAMS SYSIN statement to point to OSB.INSTALL(SPLXSEGn) or OSB.INSTALL(SPLXSGnn). This defines the data sets comprising the segment being expanded.
2. Ensure that this job only contains one step and that the member contains DELETE and DEFINE CLUSTER statements for all existing and new segment page data sets.
3. Submit the job to reallocate the page data sets.
4. Edit job OSB.INSTALL(S6A5FTMP) from Task E, Modify your segment allocations, page 94 and include //DD1,2,,,*n* DD statements for each page data set in the segment.
5. Submit the job to initialize each data set with S6BTLFPS.

Task K Restore the data to the newly formatted page data sets

1. Edit the JCL for S6BTLRPS changing the PARM='SEG=*nn*' to specify the segment being restored by S6BTLRPS.
2. Change the JOURNAL DD to point to the balanced segment backup created in Task H, Distribute the segment data across the page data sets, page 95.
3. Submit the JCL for S6BTLRPS to repopulate the segment data sets.

Task L Backup and verify the data set

This task ensures that the restore has been successful.

1. Using new backup names, execute the commands defined in Task B, Perform a backup of the segment, on page 93 and Task C, Verify the backup is good, page 93.

Make sure that you use new backup names so that previous backups are not overwritten. This is especially important for the backup created in Task B, Perform a backup of the segment, page 93 which is the fallback backup for this entire procedure.

2. Correct any detected errors before continuing.

The timestamps created by S6BBRBAL for relocated pages are expressed in UTC. If you are running your system with page time stamping "OFFSET=LOCAL" rather than "OFFSET=UTC", then you must observe the following.

- If your local time is a positive offset from UTC (you are ahead of UTC locally) then you must be very careful to purge old page images that might exist in the spin, merge and if you use continuous backup, the backup files. Restart your continuous backup from a freshly created backup using S6BTLBPS. You may bring the target segments back online immediately after using S6BBRBAL.
- If your local time is a negative offset from UTC (you are behind UTC locally) then you should not bring the target segments back online until the local time has caught up to the UTC at which S6BBRBAL ran. For instance, if your local time is -2 hours from UTC and S6BBRBAL is run at 13:00:00 UTC then you should not bring the segment back online until after 13:00:00 local time (15:00:00 UTC) is reached. Failure to do this may corrupt future continuous backups.

Refer to *TIBCO Object Service Broker for z/OS Utilities* for detailed information concerning recovery and S6BBRBAL.

Task M Shutdown the Data Object Broker

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs, if necessary.
3. Shut down the servers.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

For detailed information about shutting down TIBCO Object Service Broker, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Task N Restart the Data Object Broker

From the system console, issue the command to start the Data Object Broker. You may need to bring the expanded segment online, using DBONLINE. Example:

/F OSTARDOB, DBONLINE=*nn*

For information about starting TIBCO Object Service Broker, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Procedure 2: Using Unload and Load Utilities

Procedure 1: Using Backup and Restore Utilities, page 93 is the recommended procedure but if your segment does not have many tables you may find it more convenient to use the unload and load utilities instead.

Task A Take the segment offline, using the dboffline command

From the system console, issue the following command, replacing *nn* with the segment number:

/F OSTARDOB, DBOFFLINE=*nn*

Task B Perform a backup of the segment

Edit the JCL for S6BTLBPS (Backup Page Data Sets utility) and change the SEG#=0 parameter to indicate the segment to be backed up. Submit the job to create the backup.

Task C Verify that the backup is good

Use S6BBRPTR (Batch Pointer Check utility) to check the backup. If any errors are detected, correct the problems before continuing. Once you have obtained a clean backup, save the backup for the expansion of the segment and recovery.

1. Edit the JCL for S6BBRPTR and change PARM='SEGMENT=01,HDR' to the segment number that is being processed.
2. In the same JCL, change the //BACKUP DD statement to point to the backup created in Task B.

Task D Unload the table date from the segment

Use S6BBRULHB (Batch Unload utility) to unload the table data. For each table to be unloaded from the segment, you must prepare control statement information using the BATCHUNLD_CARDS tool. For details, see *TIBCO Object Service Broker Shareable Tools*.

Task E Increment the number of data sets

1. Edit OSB.INSTALL(S6A5DBDG) to increase the number of data sets by incrementing the ACBS=*n* parameter for the segment.
2. Submit the job OSB.INSTALL(S6A5DBDG) to activate the definition revisions.

Task F Modify your segment allocations

1. Edit the member CNTL(SPLXSEG*n*) or CNTL(SPLXSG*nn*), which describes your segment allocation:
   a. Add a DELETE statement
   b. Add a DEFINE CLUSTER statement for the new page data set being added.
2. Create a job OSB.INSTALL(S6A3ATMP) modelled after OSB.INSTALL(S6A3ALOC) and include only 1 IDCAMS step that contains the new DEFINE CLUSTER statements:

```
OSB.INSTALL(S6A3ATMP)
//*-------------------------------------------------------------------*
//*         Allocate SEGMENT nn new data sets.
//*-------------------------------------------------------------------*
//STEP4     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=$SYSPRT$
//SYSIN     DD DSN *
DEFINE CLUSTER (NAME($HLQVSAM$.$SLQ$.$SEGnnNAME$.PAGEn) UNIQUE -
 VOLUMES(HRN157) NONINDEXED RECSZ(4089,4089) CYLINDERS(20,0) CISZ(4096))
```

3. Submit OSB.INSTALL(S6A3ATMP) to delete and allocate the segment data sets.
4. Using the JCL in OSB.INSTALL(S6A5FRMT) as a model, create a member OSB.INSTALL(S6A5FTMP) to format the segment's data sets using utility S6BTLFPS. Include a DD statement for each new page data set within the segment:

```
OSB.INSTALL(S6A5FTMP)
     //*-------------------------------------------------------------*
     //*         Format SEGMENT nn data sets                         *
     //*-------------------------------------------------------------*
     //FRMTSEGS EXEC PGM=S6BTLFPS
     //STEPLIB  DD DSN=$HLQNONV$.$INSTVER$.AUTH,
     //         DISP=SHR
     //DBDLIB   DD DSN=$HLQVSAM$.$SLQ$.DBDLIB,
     //         DISP=SHR
     //SYSPRINT DD SYSOUT=$SYSPRT$
     //*        INCLUDE ONLY THE NEWLY ALLOCATED PAGE DATASET(S) HERE
     //DD1      DD DISP=SHR,DSN=$HLQVSAM$.$SLQ$.$SEGnnNAM$.PAGEn
     ...
     /*
```

Task G Reload each table's data into the expanded segment

Use S6BBBRTBL (Batch Load utility) to load the table data. For each table to be loaded from the segment, prepare control statement information using the BATCHLOAD_CARDS tool. For details, see *TIBCO Object Service Broker Shareable Tools*.

Task H Perform a backup of the expanded segment

Edit the JCL for S6BTLBPS (Backup Page Data Sets utility) and change the SEG#=0 parameter to indicate the segment to be backed up. Submit the job to create the backup.

Task I Verify that the backup is good

Use S6BBRPTR (Batch Pointer Check utility) to check the backup. If errors are detected, correct the problems before continuing. Once you have obtained a clean backup, save the backup for the expansion of the segment and recovery.

1. Edit the JCL for S6BBRPTR and change PARM='SEGMENT=01,HDR' to the segment number that is being processed.
2. In the same JCL, change the //BACKUP DD statement to point to the backup created in Task H, Perform a backup of the expanded segment, page 100.

Task J Shutdown the Data Object Broker

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs, if necessary.
3. Shut down the servers.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

For details, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Task K Restart the Data Object Broker

From the system console, issue the command to start the Data Object Broker. You may need to bring the expanded segment online, using DBONLINE. Example:

/F OSTARDOB, DBONLINE=*nn*

For details, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Chapter 9 Configuring Your Pagestore on Open Systems

This chapter describes how to configure the TIBCO Object Service Broker Pagestore on open systems.

Topics

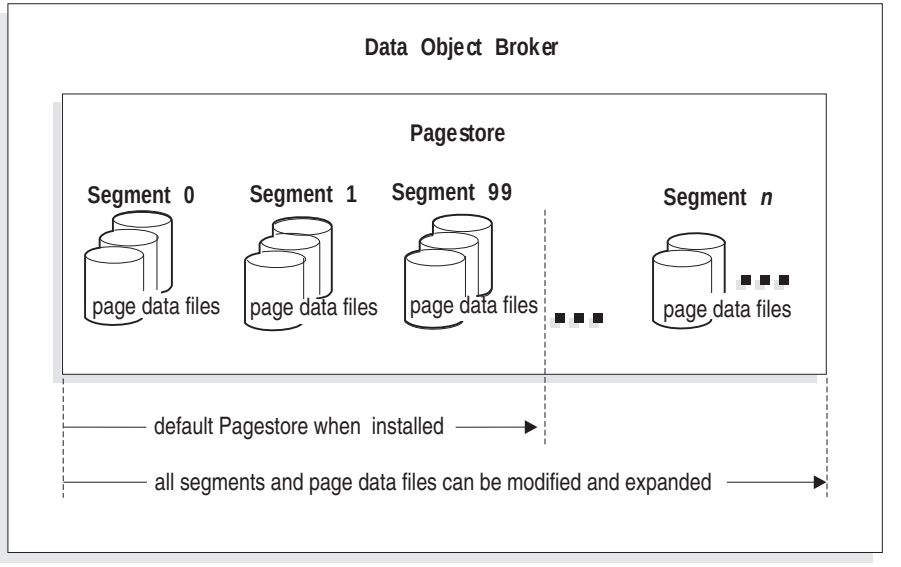
TIBCO Object Service Broker Data Store Architecture

The Data Object Broker accesses data maintained in the TIBCO Object Service Broker data store, called the Pagestore, which is divided into segments. The Pagestore is the repository for all segments associated with a specific Data Object Broker.

With TIBCO Object Service Broker, you can isolate or group your data and store it in separate segments of the Pagestore, according to the use and sensitivity of the data. With your Pagestore divided, you can also take one segment offline, back it up, clear the data, and reload it while keeping all other applications online. As your database grows, your Pagestore can be enlarged or additional segments can be added as needed.

Relationships

The following illustration shows the relationship of the Data Object Broker, the Pagestore, and the segments within the Pagestore.



TIBCO Object Service Broker Pagestore

How is TIBCO Object Service Broker Installed?

With TIBCO Object Service Broker installed, your Pagestore consists of a *minimum* of three segments:

Segment 0 — This base segment contains the MetaStor, made up of definitions, characteristics, access paths, and storage locations of all data and programs.

Segment 0 contains one page file with 5000 pages when it is distributed.

Segment 1 — This segment helps you separate your user data from the dictionary information stored in segment 0.

Segment 1 contains three page files when it is distributed.

Segment 99 — This segment is used to hold security audit data.

What is the Logical View of the Data?

The logical view of the data is stored in the form of tables. The data for a table resides completely within a specific segment and can span all the physical files comprising the segment. The definition of the table is stored in segment 0, which is known as the MetaStor.

How is the Data Physically Stored?

Data is ultimately stored within page files as 4096 byte pages. Each 4 KB page has a header record that identifies:

- The segment to which it belongs
- The page file within the segment
- The page number and type
- Page chaining information
- The data size and number of occurrences
- Date and time of the last update to this page
- Checkpoint/transaction information of the update

The pages are written to the page files by the Data Object Broker as part of checkpoint processing.

What Changes Can be Made to the Pagestore?

Changes in the requirements of your test or production TIBCO Object Service Broker systems can require you to increase or decrease the amount of free space in the Pagestore. Free space can be modified in the Pagestore in three ways:

- You can increase or decrease the size of existing Pagestore files.
- You can partition your Pagestore by adding additional segments.
- You can add additional files to an existing segment.

Each data table in a TIBCO Object Service Broker system resides on a particular segment and the segment number is associated with the table. You can move the segment as your system needs change.

You can also restore the Pagestore or segments of the Pagestore from backup.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* for more information about restoring the Pagestore from backup.

Modifying the Size of an Existing Page File

As your system requirements change, you can increase or decrease the size of an existing page file using utilities that are supplied with TIBCO Object Service Broker. These utilities backup your data, allocate new files, and restore your data to the new files.

This process does not re-organize your file. Each page file has the same number of used pages after the restore and each page is restored to its original location in the page file.

Ensure that the resized page files are at least large enough to accommodate the highest page number in the backup file that is to be restored.

Increasing or Decreasing the Size of a Page file

To change the size of a file, complete the following steps:

1. Shut down your TIBCO Object Service Broker system or take the segment offline if it is not segment 0.
2. Take a backup of the segment whose files are to be modified, using hrntlbps.
3. Run the pointer checker utility hrnbrptr against the backup.
4. If hrnbrptr reports problems, address the errors before proceeding.
5. If necessary, repeat the three previous steps until no problems are reported.

   Valediction of the integrity of your files can require more than one run of hrnbrptr.
6. Delete the old files.
7. Format the new page files using hrntlfps.
8. Restore the previous clean backup using hrntlrps.
9. Take a full segment backup and run it through hrnbrptr.
10. Merge the last backup into the continuous backup process, if it is being used.
11. Restart the Data Object Broker or bring the segment back online.

See Also *TIBCO Object Service Broker for Open Systems Utilities* and *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* for specific information about the utilities mentioned.

Defining an Additional Segment

Tasks Required to Define Additional Segments

You can define additional segments as your system requirements increase. To define an additional segment, complete the following tasks:

Task A Edit and save the sample file

Complete the following steps:

1. Edit the sample dbdef file in the database folder under your TIBCO Object Service Broker install folder.
2. Locate the "DB TYPE=PAGE,ACBS=*n*,…" statement.
3. Replicate this statement.
4. Replace the values. Refer to *TIBCO Object Service Broker for Open Systems Installing and Operating* for information about the dbdef file.
5. Save the edited version of the file.

Sample File

The sample file shown here defines five segments. The first two, segment 0 and segment 1, use the installation supplied names METASTOR and SEG01. The three additional page segments are named SALES, EXPENSES, and PURCHASE.

```
DB TYPE=PAGE,ACBS=1,NAME=METASTOR,INIT=YES
DB TYPE=PAGE,ACBS=1,NAME=SEG01,ID=1,INIT=YES
DB TYPE=PAGE,ACBS=3,NAME=SALES,ID=2
DB TYPE=PAGE,ACBS=3,NAME=EXPENSES,ID=3
DB TYPE=PAGE,ACBS=3,NAME=PURCHASE,ID=4
DB TYPE=REDOLOG,NAME=REDO
DB TYPE=JOURNAL,ACBS=2,NAME=JOURNAL
DB TYPE=CLOG,NAME=CLOG
```

Task B Create a new folder

In the database folder, create a folder to contain the page files or partitions. The name of the folder must be the segment name, for example:

In Windows:

```
cd %OS_ROOT%\database
mkdir SALES
```

In Solaris:

```
cd ${OS_ROOT}/database
mkdir SALES
```

The files are created at initialization.

Task C Initialize the files for data

After you define a new folder, you must initialize its files to prepare them for data. For each new page file or partition, run the utility hrntlfps, using the following syntax:

In Windows:

```
hrntlfps %OS_ROOT%\database\PAGE4
```

In Solaris:

```
hrntlfps ${OS_ROOT}/database/PAGE4
```

Task D Shut Down the TIBCO Object Service Broker system

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs if necessary.
3. Shut down the servers that are running.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

For details, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Task E Restart the Data Object Broker

For details, see to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Adding a Page File to a Segment

If a segment is nearing its data capacity, you can use one of two procedures to add one or more page data sets to the segment:

1. Use backup and restore utilities along with segment balance. This is the recommended method especially if the segment has many tables.
2. Use unload and load utilities.

See Also *TIBCO Object Service Broker for Open Systems Utilities* and *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* for specific information about the utilities mentioned in the following sections.

Procedure 1: Using Backup and Restore Utilities

Complete the following tasks to add a page data set to an existing segment.

Task A If the segment is SEG0, shutdown the Data Object Broker

Issue the following command, replacing *nn* with the segment number:

hrncr dboffline=*nn*

Task B Perform a backup of the segment

Use hrntlbps (Backup Pagestore utility) to back up the segment. Issue the following command, replacing *nn* with the segment number:

hrntlbps -s *nn* seg*nn*.bak

Task C Verify that the backup is good

Use hrnbrptr (Batch Pointer Check utility) to check the backup.

1. Issue the following command, replacing *nn* with the segment number:

   hrnbrptr -s *nn* -h -A s*nn*Audit.txt -H s*nn*Header.txt -E s*nn*Error.txt -R s*nn*Reflog.txt -O s*nn*Orphan.txt seg*nn*.bak

2. If any errors are detected, correct the problems before continuing.
3. Save a copy of the backup to an alternative location in case it is needed for recovery at a later stage in the process.

Task D Increment the number of data files

1. Edit the dbdef file in the database folder under your TIBCO Object Service Broker install folder.
2. Locate the "DB TYPE=PAGE,ACBS=*n*,…" statement for the segment you want to expand.
3. Increment the number of data files by modifying the ACBS=*n* statement.

   Refer to *TIBCO Object Service Broker for Open Systems Installing and Operating* for information about the dbdef file.
4. Save the edited version of the file.

Task E Initialize the new page files

Use hrntlfps (Format Pagestore utility) to initialize the new page files. Issue the following command, replacing *nn* with the segment and page numbers:

- For Windows:

  hrntlfps %OS_ROOT%\database\SEG*nn*\PAGE*nn*
- For Solaris:

  hrntlfps ${OS_ROOT}/database/SEG*nn*/PAGE*nn*

Task F Perform a backup of the expanded segment

Use hrntlbps to backup the expanded segment. Issue the following command, replacing *nn* with the segment number:

hrntlbps -s *nn* seg*nn*.bak2

Task G Verify the expanded data set

Use hrnbrptr (Batch Pointer Check utility) to check the backup.

1. Issue the following command, replacing *nn* with the segment number:

   hrnbrptr -s *nn* -h -A s*nn*Audit.txt -H s*nn*Header.txt -E s*nn*Error.txt -R s*nn*Reflog.txt -O s*nn*Orphan.txt seg*nn*.bak
2. If any errors are detected, correct the problems before continuing.
3. Save a copy of the backup to an alternative location in case it is needed for recovery at a later stage in the process

At this point the expanded segment can be used by the Data Object Broker by skipping to Shutdown the Data Object Broker, page 111; however, we recommend that hrnbrbal (Segment Balance utility) be used to balance the data across all the segment's page data sets. This will distribute I/O for new data across all page data sets, yielding better performance.

Task H Distribute the segment data across the page data sets

hrnbrbal (Segment Balance utility) takes your backup and creates a new backup data file reflecting evenly populated page files. To use this utility issue a command similar to the following replacing *nn* with the segment number and *xx* with the number of files in the new segment definition:

hrnbrbal -A auditreport -p 40000 -s *nn* -f *xx* -N seg*nn*new.bak seg*nn*old.bak

Refer to *TIBCO Object Service Broker for Open Systems Utilities* for detailed information about using this utility.

Task I Verify the balanced segment

Use hrnbrptr (Batch Pointer Check utility) to check the backup.

1. Issue the following command, replacing *nn* with the segment number:

   hrnbrptr -s *nn* -h -A s*nn*Audit.txt -H s*nn*Header.txt -E s*nn*Error.txt -R s*nn*Reflog.txt -O s*nn*Orphan.txt seg*nn*.bak

2. If any errors are detected, correct the problems before continuing.
3. Save a copy of the backup to an alternative location in case it is needed for recovery at a later stage in the process.

Task J Reallocate and format the page date files

Use hrntlfps to reallocate and format the new page files in the segment. Issue the following command, replacing *nn* with the segment and page numbers:

- For Windows:

  hrntlfps %OS_ROOT%\database\SEG*nn*\PAGE*nn*

- For Solaris:

  hrntlfps ${OS_ROOT}/database/SEG*nn*/PAGE*nn*

Task K Restore the data to the newly formatted page files

Use hrntlrps (Restore Pagestore utility) to restore the data to the newly formatted page files, i.e., the backup created in Task H, Distribute the segment data across the page data sets, on page 110. Issue the following command, replacing *nn* with the segment number:

hrntlrps -r -s *nn* segnew*nn*.bak

Task L Backup and verify the data set

This task ensures that the restore has been successful.

1. Using new backup names, execute the commands defined in Perform a backup of the segment, page 108 and Verify that the backup is good, page 108.
2. Correct any detected errors before continuing.

Make sure that you use new backup names so that previous backups are not overwritten. This is especially important for the backup created in Perform a backup of the segment, page 108 which is the fallback backup for this entire procedure.

Task M Shutdown the Data Object Broker

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs, if necessary.
3. Shut down the servers.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

For details, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Task N Restart the Data Object Broker

From the system console, issue the command to start the Data Object Broker. You may need to bring the expanded segment online, using dbonline, which is included in the following example:

hrncr dbonline=*nn*

For details, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Procedure 2: Using Unload and Load Utilities

Procedure 1: Using Backup and Restore Utilities, page 108 is the recommended procedure but if your segment does not have many tables you may find it more convenient to use the unload and load utilities instead.

Task A If the segment is SEG0, shutdown the Data Object Broker

Issue the following command, replacing *nn* with the segment number:

hrncr dboffline=*nn*

Task B Perform a backup of the segment

Use hrntlbps (Backup Pagestore utility) to back up the segment. Issue the following command, replacing *nn* with the segment number:

hrntlbps -s *nn* seg*nn*.bak

Task C Verify that the backup is good

Use hrnbrptr (Batch Pointer Check utility) to check the backup.

1. Issue the following command, replacing *nn* with the segment number:

   hrnbrptr -s *nn* -h -A s*nn*Audit.txt -H s*nn*Header.txt -E s*nn*Error.txt -R s*nn*Reflog.txt -O s*nn*Orphan.txt seg*nn*.bak

2. If any errors are detected, correct the problems before continuing.
3. Save a copy of the backup to an alternative location in case it is needed for recovery at a later stage in the process.

Task D Unload the table date from the segment

Use hrnbrulb (Batch Unload utility) to unload the table data. For details, see *TIBCO Object Service Broker for Open Systems Utilities*.

For each table to be unloaded from the segment, you must prepare control statement information using the BATCHUNLD_CARDS tool. For details, see *TIBCO Object Service Broker Shareable Tools*.

Task E Increment the number of data files

1. Edit the dbdef file in the database folder under your TIBCO Object Service Broker install folder.
2. Locate the “DB TYPE=PAGE,ACBS=*n*,…” statement for the segment you want to expand.

3. Increment the number of data files by modifying the ACBS=*n* statement.

   Refer to *TIBCO Object Service Broker for Open Systems Installing and Operating* for information about the dbdef file.

4. Save the edited version of the file.

Task F Initialize the segment page files

Use hrntlfps to initialize the new page files in the segment. Issue the following command, replacing *nn* with the segment and page numbers:

- For Windows:

  hrntlfps %OS_ROOT%\database\SEG***nn***\PAGE*nn*

- For Solaris:

  hrntlfps ${OS_ROOT}/database/SEG*nn*/PAGE*nn*

Task G Reload each table's date into the expanded segment

Use hrnbrtbl (Batch Load utility) to load the table data. For details, see *TIBCO Object Service Broker for Open Systems Utilities*.

For each table to be unloaded from the segment, you must prepare control statement information using the BATCHLOAD_CARDS tool. For detailed information about this tool refer to *TIBCO Object Service Broker Shareable Tools*.

Task H Perform a backup of the segment

Use hrntlbps to backup the expanded segment. Issue the following command, replacing *nn* with the segment number:

hrntlbps -s *nn* seg*nn*.bak2

Task I Verify that the backup is good

Use hrnbrptr (Batch Pointer Check utility) to check the backup.

1. Issue the following command, replacing *nn* with the segment number:

   hrnbrptr -s *nn* -h -A s*nn*Audit.txt -H s*nn*Header.txt -E s*nn*Error.txt -R s*nn*Reflog.txt -O s*nn*Orphan.txt seg*nn*.bak

2. If any errors are detected, correct the problems before continuing.

3. Save a copy of the backup to an alternative location in case it is needed for recovery at a later stage in the process.

Task J Shutdown the Data Object Broker

To ensure an orderly shutdown, complete the following steps:

1. Determine if users are logged in to the system.
2. Cancel user IDs, if necessary.
3. Shut down the servers.
4. Shut down the Execution Environments.
5. Shut down the Data Object Broker.

For details, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Task K Restart the Data Object Broker

From the system console, issue the command to start the Data Object Broker. You may need to bring the expanded segment online, using dbonline, which is included in the following example:

hrncr dbonline=*nn*

For details, see *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Chapter 10 Re-allocating the Segment Number

This chapter describes how to re-allocate segment numbers.

Topics

Changing the Segment Number of a Table

You can decide how to allocate the tables to various segments based upon your system requirements. For example, you decide to relocate all sensitive tables that require daily backup to a separate segment so that you can take that segment offline and back it up more frequently than other data. Based on these requirements, it can be necessary to change the segment number of a table.

When user tables are defined, the definitions are stored on segment 0 and the table data is stored on the default TDS segment as specified in the creator's user profile. Users of the table do not need to know how or where it is physically stored.

Conditions for Changing a Segment Number

The following conditions apply to changing the segment number of a table:

- The table type must be TDS.
- The table whose segment number is being changed cannot be a TIBCO Object Service Broker system table (that is, tables with creator HURON).
- The target segment must be online.
- The table specified must be empty.

How Do You Change A Segment Number?

You can change the segment number:

- Interactively, using the Table Definer from the administrator's workbench
- From a rule, by calling the MOVTAB tool

See Also *TIBCO Object Service Broker Managing Security* for information about user profiles.

Procedure to Change a Segment Number

Changing the segment number consists of the following tasks:

1. Unload or Clear the Data, page 117
2. Change the segment number, page 117
3. Reload the table, page 119

These tasks are described in the following sections.

Task A Unload or Clear the Data

If the table contains data, use the following table to determine the method you should use to unload or clear the data from the table:

Action	Using a Tool	Using a Batch Process
Unload the data.	UNLOAD	S6BBRULH/hrnbrulh or S6BBRULB/hrnbrulb
Clear the data.	$CLRTAB	S6BRCLR/hrnbrclr

Task B Change the segment number

The following methods are available to change the segment number:

- Define Table option from the administrator's workbench
- The MOVTAB tool supplied by TIBCO Object Service Broker

Using the Define Table Option

To use the Define Table option, complete the following steps:

1. On the administrator's workbench, position the cursor beside the DT Define Table option and press Enter.

A list of tables similar to the following illustration appears.

```
                          List of defined tables
 Command ==>                                                      Scroll P
            Enter one or more line commands or a primary command
          NAME        TYPE CREATOR    CREATD     UNIT    MODIFIER   MODIFD    BI*
    ---------------- ---- -------- ---------- -------- -------- ---------- ---
  _ ABC1             TDS  USR40    2000-04-07 USR40    USR40    2000-04-07 N
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH   SEARCH   2000-04-06 N
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10    ABC10    2000-04-03 N
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH   SEARCH   2000-03-28 Y
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR   RPSUSR   2000-03-09 N
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR   RPSUSR   2000-03-06 N
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 Y
  _ DATATEST         TDS  RPSUSR   2000-03-06 SEARCH   SEARCH   2000-03-06 N
  _ NRGTEST          TDS  USR40    2000-03-06 USR40    USR40    2000-03-06 N
  _ USRWEB           TDS  USR00    2000-03-06 USR40    USR40    2000-03-06 N
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL USR50    2000-03-05 N
B-Bind  C-Clear  D-Delete  M-Move  P-Prt  R-Reset Bind  S-Sel  X-SIX Bld/Del
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

2. Type the line command **M** beside the required table.
3. Press Enter.

   The following screen appears to enter a new segment number.

```
To complete this command:
         NAME        TYPE CREATOR    CREATD     UNIT    MODIFIER   MODIFD    BI*
   ---------------- ---- -------- ---------- -------- -------- ---------- ---
 M @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL USR50    2000-03-05 N
Enter parameter(s): Destination segment number


 SEGMENT#           ===>




PFKEYS: ENTER=PROCESS 3=PROCESS 12=CANCEL
```

4. Type the new segment number next to the field 'SEGMENT# ===>'
5. Press Enter or PF3 to process the command.

Using MOVTAB

To use the TIBCO Object Service Broker supplied MOVTAB tool, execute it from a rule by specifying:

```
CALL MOVTAB(tablename, segmentID);
```

where

tablename	Is the name of the empty table whose segment number you are changing
segmentID	Is the new segment number to be used for the table

Task C Reload the table

If you unloaded the data from your table after changing the segment number of the table, use the LOAD tool to reload the data. For a large table, you can use the batch utility S6BBRTBL/hrnbrtbl.

See Also *TIBCO Object Service Broker Shareable Tools* for more information about MOVTAB, UNLOAD and $CLRTAB.

Utilities for your operating environment for more information about the S6BBRTBL/hrnbrtbl, S6BBRULH/hrnbrulh, S6BBRULB/hrnbrulb, and S6BBRCLR/hrnbrclr batch utilities.

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W