TIBCO® Object Service Broker for Open Systems

Installing and Operating

Software Release 6.0
July 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

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

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

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

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

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

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

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

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

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

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

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

Contents

Preface

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

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

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Object Service Broker Documentation

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

Fundamental Information

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

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

Application Development and Management

The following manuals provide information about application development and management:

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

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

System Administration on the z/OS Platform

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

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

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

System Administration on Open Systems

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

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

External Database Gateways

The following manuals describe external database gateways:

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

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

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
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 Installing TIBCO Object Service Broker

This document explains how to install TIBCO Object Service Broker on Microsoft Windows and Solaris systems.

Topics

Installer Overview

This section summarizes the installation process.

Installation Modes

The installer allows you to run in different modes.

- **GUI Mode** In GUI mode, the installer presents panels that allow you to make choices about product selection, product location, and so on. When you invoke the installer by double-clicking on the icon, GUI mode is used.
- **Console Mode** Console mode allows you to run the installer from the command prompt or terminal window. This is used primarily for non-Windows environments.
- **Silent Mode** Silent mode installs without prompting you for information. To use this mode, you must first generate a *response file* (using GUI mode or Console mode) that contains the input values you want to use for the installation.

For details, see Choosing an Installation Mode on page 8 (Windows) or Choosing an Installation Mode on page 19 (Solaris).

Base Components and SDK Clients

The TIBCO Object Service Broker installation is a two part process. The first part installs the base TIBCO Object Service Broker binary and database components (including the MetaStor). The second part, optionally launched from the first process, installs the TIBCO Object Service Broker SDK Clients. These clients are:

- TIBCO Object Integration Gateway: J2EE for Windows and Solaris.
- For Windows only: TIBCO Object Integration Gateway COM and .NET components and TIBCO Object Service Broker Adapter for JDBC-ODBC. The C++ and Java SDKs are part of the base components for all platforms.

Optional TIBCO Products for TIBCO Object Service Broker

Depending on the tasks you wish to perform, you can install one or more other TIBCO Software products.

Table 3 Optional TIBCO Products

Product	Purpose
TIBCO Service Gateways 6.0	TIBCO Service Gateways provide interfaces to external databases, such as IMS and DB2. These products are available as separate installations and can be installed after installing TIBCO Object Service Broker. For details, see *TIBCO Service Gateway* for the desired server interface.
TIBCO EMS	The TIBCO Object Service Broker EMS interface enables direct integration with TIBCO Enterprise Message Service.
TIBCO Hawk and TIBCO Mainframe Service Tracker	On Open Systems, you can monitor systems with TIBCO Hawk. On z/OS, TIBCO Mainframe Service Tracker enables integration with TIBCO Hawk.
TIBCO BusinessWorks OSB Plug-in	The TIBCO BusinessWorks OSB Plug-in contains resources for retrieving or updating TIBCO Object Service Broker data for use in a TIBCO BusinessWorks process. This product is available as a separate installation and can be installed after installing TIBCO Object Service Broker. For details, see *TIBCO BusinessWorks OSB Plug-in User's Guide*.

Preparing for Installation

This section provides an overview of the installation process.

Prerequisite Software

For supported software versions, see the readme.

The installers require a JRE/JDK. When prompted by the installer, specify the location of the JRE/JDK.

For Microsoft Windows, the SDK Clients installer requires that you install the supported Microsoft .NET Framework package, and the Microsoft Visual J# Redistributable Package.

Installer Disk Space Requirements

- **Microsoft Windows** The entire package is extracted into a temp folder, typically `SystemDrive:\Temp` or `SystemDrive:\Documents and Settings\<user_name>\Local Settings\Temp`. The installer requires 165MB of free space in the temp directory.
- **Solaris Platforms** When a regular (nonroot) user installs a TIBCO Software product, the installation registry is maintained in the user's home directory. As more products are installed, entries are added. The user's home directory must at least have 100 MB of free disk space.
- **Directory for Installers** Ensure that the downloaded installer files for both the Base and the SDK Clients are located in a user-writable directory and that at least 165 MB of space is available.

Installer Account

You can install TIBCO Object Service Broker on either Microsoft Windows or Solaris. However, you must have administrator privileges for the machine on which to install the product. Otherwise, the installer exits. In that case, log out of the system and log in as a user with the required privileges, or request your system administrator to assign the privileges to your account.

To enable TIBCO Object Service Broker programs to run properly in a Windows 2008 system, the setting for User Account Control (UAC) must be off.

To install the product on a network drive, ensure that the installation account has permission to access that drive.

Windows Terminal Server

Windows Terminal Server contains two modes: `Execute` and `Install`. By default, all users are logged in in `Execute` mode, which enables them to run the applications. To install TIBCO Object Service Broker for all users, change to `Install` mode.

The best way to install TIBCO Object Service Broker is through the Add/Remove Programs Control Panel applet, which automatically sets the mode to `Install` during installation and then back to `Execute` on completion.Alternatively, do the following:

1. Manually change the mode to `Install`. Type:

   ```
   C:\> change user /install
   ```

2. Change back to `Execute` mode. Type:

   ```
   C:\> change user /execute
   ```

3. Check your current mode. Type:

   ```
   C:\> change user /query
   ```

Depending on whether you install in `Execute` mode or `Install` mode, the installation registry is in your user home directory or in the folder `%SystemRoot%\WINDOWS`, respectively.

Solaris

You can install TIBCO Object Service Broker as a regular (nonroot) user or superuser (root). Different users can install the same product at different locations.

Installation Parameters

When you install the software (regardless of the installation mode used), you must specify the installation parameters listed below. Determine values appropriate for your site for these parameters before initiating the installation.

Parameter	Description
DOB Nodename	Name of the TIBCO Object Service Broker database engine. This uniquely identifies the TIBCO Object Service Broker system.

Parameter	Description
DOB Alias	If the name of the database engine is greater than 8 characters, provide the alias name. Use a unique name that is no more than 8 characters.
DOB Port	Listening port for the database engine. Use a unique number between 1024 and 65535.
DOB IPCKEY (Solaris only)	IPC key value for the database engine. Use a unique hexadecimal value between 0 and 0x7FFF.
osMon Port	Listening port for the Execution Environment. Use a unique number between 1024 and 65535.
TN3270 Port	Listening port for the customer-supplied 3270 emulator. Use a unique number between 1024 and 65535.
osMon IPCKEY (Solaris only)	IPC key value for the Execution Environment. Use a unique hexadecimal value between 0 and 0x7FFF.

JDBC Deployment Parameters

You must also specify the JDBC deployment parameters listed in the table below.

Parameter	Description
Path to Server Folder in JDK's JRE	Full path to the server folder in JDK's JRE.
Hostname	Name of the host where JDBC Services will be run.
SQL Agent Port	Used in Open Access Administration Configuration. Defaults to 19985.
SQL Service Port	Used to communicate with Data Broker. Defaults to 19988.

Installation Verification Procedure Parameters

The installation of the TIBCO Object Service Broker SDK Clients includes an optional Installation Verification Procedure (IVP); to make use of it, the following parameters are required:

Parameter	Description
Host Name	Host where an osMon is active to serve the IVP requests.
Port Number	The osMon port number.
OSB Userid	The Userid to use in each IVP request.
Password	The password associated with the Userid.
Java Home	The JRE/JDK path to be used by the IVP process.
SQL Service Host	The host where the SQL services are running, which will be used by the JDBC and ODBC 64-bit Adapter installation.
SQL Service Port	The port used by the SQL services required for the JDBC and ODBC 64-bit Adapter installation.

If the IVP is invoked as part of a TIBCO Object Service Broker base component installation, all the parameters will be populated with the proper information, with the exception of the Password field, which must be typed in. If the IVP is invoked from an SDK only installation, the parameters may be filled in, but will require reviewing

Installing on Microsoft Windows

Before starting the installation procedure, ensure that your system meets the hardware and software requirements, and that you have reviewed the pre-installation steps.

Installation Files

Installation file names for TIBCO Object Service Broker vary by version number and platform, using the following general format:

`TIB_osb_`*`version_platform`*

where *version* is the three-digit version number for this TIBCO Object Service Broker release and*platform* is an abbreviated form of the hardware platform for which the executable is intended.

The installation files on Windows for this release are the following:

```
TIB_osb_6.0.0_win_x86_64.exe       (Base Product)
TIB_osbC_6.0.0_win_x86_64.exe   (SDK Clients)
```

Download the product from the TIBCO Software web site by following these steps:

1. Contact TIBCO Software for a password, directory information, and so on.
2. Connect to the TIBCO Software web site with the required information.
3. Download the installation file appropriate for your platform.

Choosing an Installation Mode

Table 4 describes the modes for installing the product on Microsoft Windows.

Table 4 Installation Modes on Windows

Installation Mode	Description
GUI mode	Allows you to input values in panels. Use the following command: `TIB_osb_6.0.0_win_x86_64.exe`
Console mode	Allows you to install the software without using a GUI. The installer will prompt you for values. Use the following command: `TIB_osb_6.0.0_win_x86_64.exe -console`

Table 4 Installation Modes on Windows

Installation Mode	Description

Silent mode	To use this mode, you must first generate a response file (using GUI mode or Console mode) that contains the input values you want to use for the installation. You generate a response file using the following command:
	`<installer> -options-record <responseFileName>`
	where <installer> is the installer executable (or script file on Solaris), and *responseFileName* is the name of the response file to be generated. For an example, see "Install and Generate a Response File" below.
	Type the following at the command prompt to use this mode:
	`TIB_osb_6.0.0_win_x86_64.exe -silent -options <responseFileName>`

Install and Generate a Response File	You can generate a response file during installation which you can later use to invoke the installer with the selected values as default values (GUI mode) or as selected values (silent mode).
	Add the following statements at the top of the response file so that the silent process takes the proper responses:
	`-G replaceExistingResponse=yesToAll`
	`-G replaceNewerResponse=yesToAll`
	To install and generate a response file, use the following command:
	`TIB_osb_6.0.0_win_x86_64.exe -options-record <responseFileName>`

GUI or Console mode Install Using a Response File	You can use a previously generated response file for installation. For non-silent modes, the response file determines the defaults that are presented. For silent mode, the response file determines what will be installed. To install using a response file, use the following command: `TIB_osb_6.0.0_win_x86_64.exe -options <responseFileName>` (GUI mode) `TIB_osb_6.0.0_win_x86_64.exe -console -options <responseFileName>` (Console mode)

Table 4 Installation Modes on Windows

Installation Mode	Description
Combining Options	You can combine the different available options. For example, to install using Console mode and generate a response file, use: `TIB_osb_6.0.0_win_x86_64.exe -console -options-record <responseFileName>`

Installing the Software

The following procedure is for a new installation. For instructions on upgrading from a previous release, see the section titled "Migration from Previous Releases on Open Systems" in *TIBCO Object Service Broker Release Notes*.

Before proceeding, you should have determined which components you will install as described in Base Components and SDK Clients on page 2. If you will install the SDK Clients, ensure that the SDK Clients installer file has been placed in the same folder as the base components installer, and that this folder is writable (*before* invoking the base product installer). Approximately 5 KB of available space is needed.

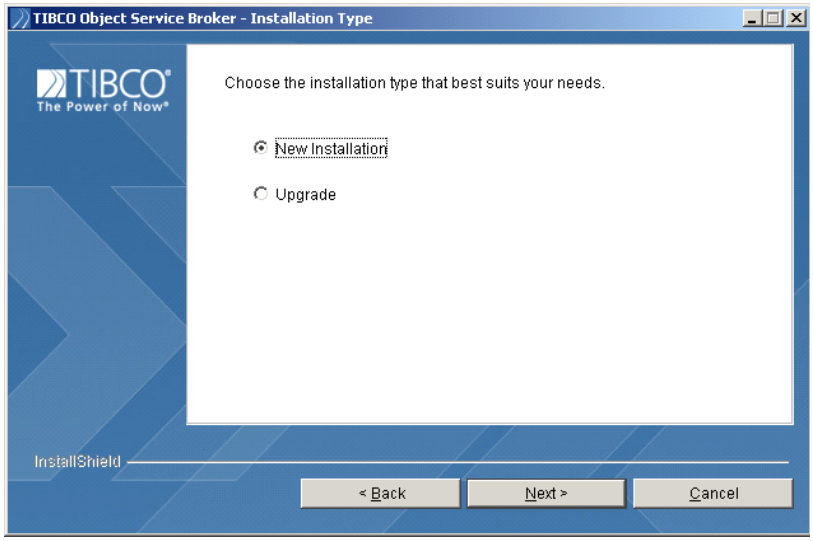
Base Components Installation (with or without SDK Clients installation)

To install TIBCO Object Service Broker, perform the following:

1. Initiate installation by issuing the command of the desired installation mode.
2. When prompted, accept the license agreement.

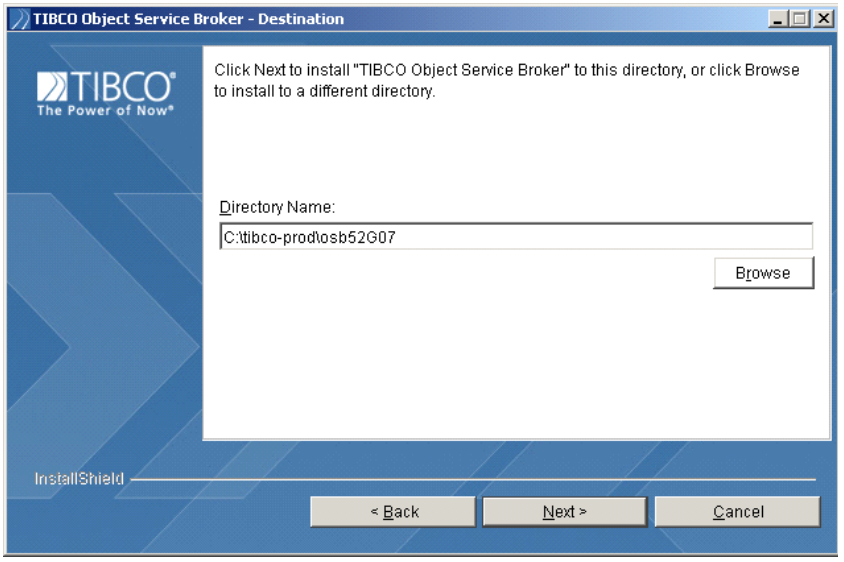
If this is the first time you install a TIBCO product, a prompt or panel requests the name of a directory for *TIBCO_HOME*, in which you could install all TIBCO products. However, you need not install OSB under the *TIBCO_HOME* umbrella.

3. When prompted, select New installation. In GUI mode, the following panel displays for this choice:



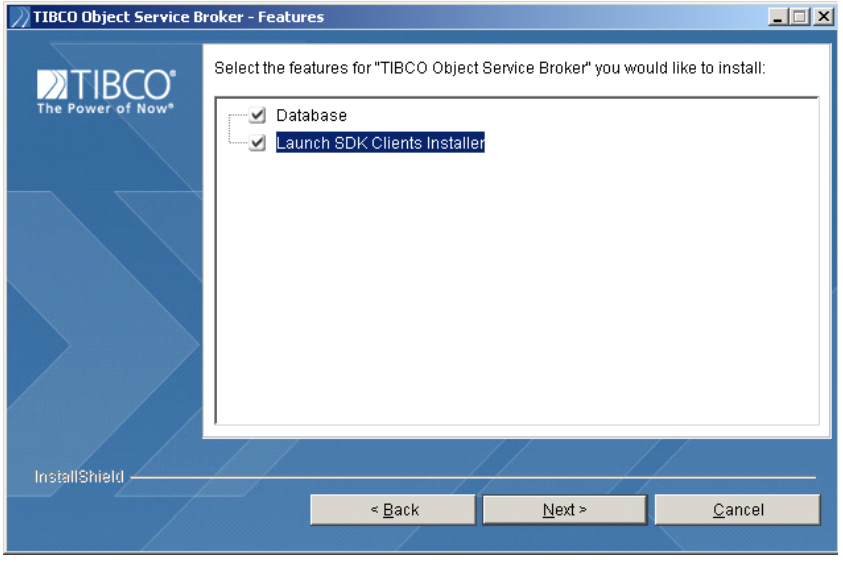
Use the upgrade selection if you would like to bring up an OSB system at release 5.0 or 5.2 to the current 6.0 version level. The HURON enviroment variable should point to the target system to be upgraded.

4. Specify the installation directory where the product will be installed. In GUI mode, the following panel displays:

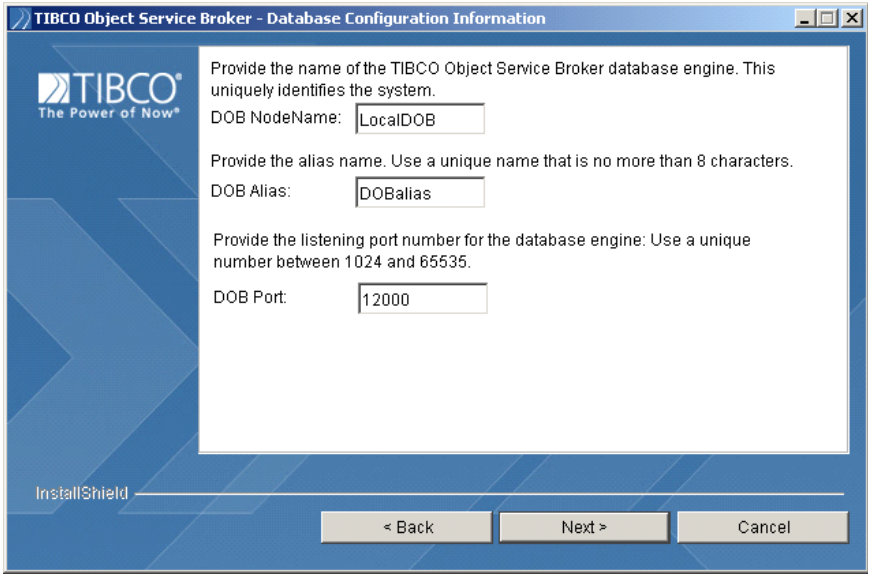


On Microsoft Windows, the default installation directory is `c:\tibco`.

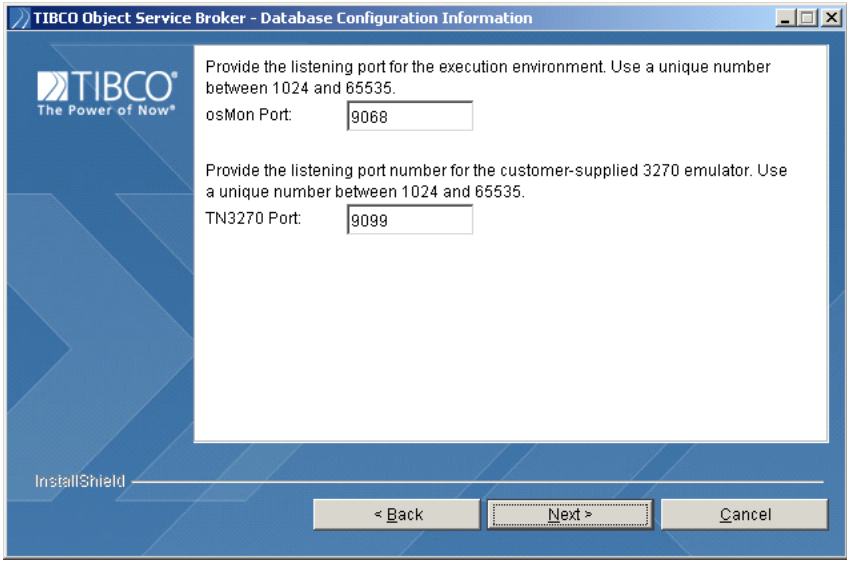
5. Select the components to install. If you wish to install SDK Clients, specify that the SDK Clients installer should be launched. In GUI mode, the following panel displays for these selections:



6. Specify the database engine parameters (described in Installation Parameters). In GUI mode, the following panel displays:

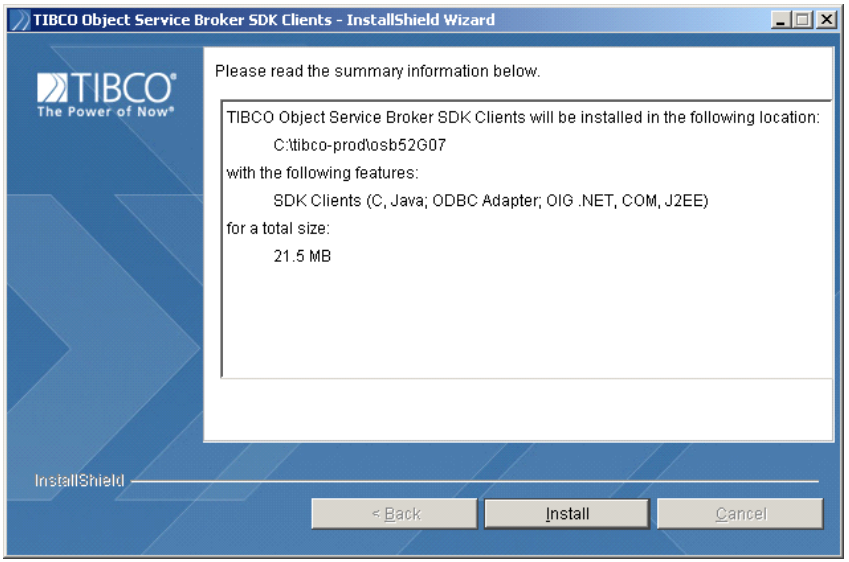


7. Specify the system parameters (described in Installation Parameters). In GUI mode, the following panel displays:



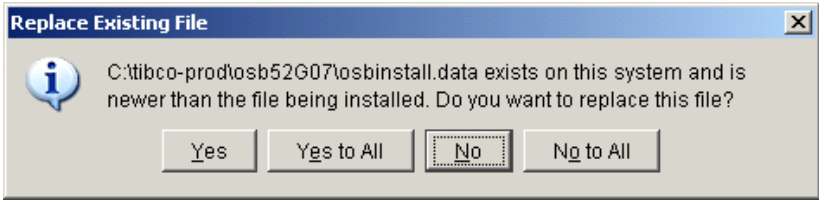
8. When a summary of your installation selections displays, click the **Install** button.

9. When the base component installation finishes, the SDK Clients installer is automatically launched. In GUI mode, the panel below displays. If you did not choose to install SDK Clients, go to Step 13.



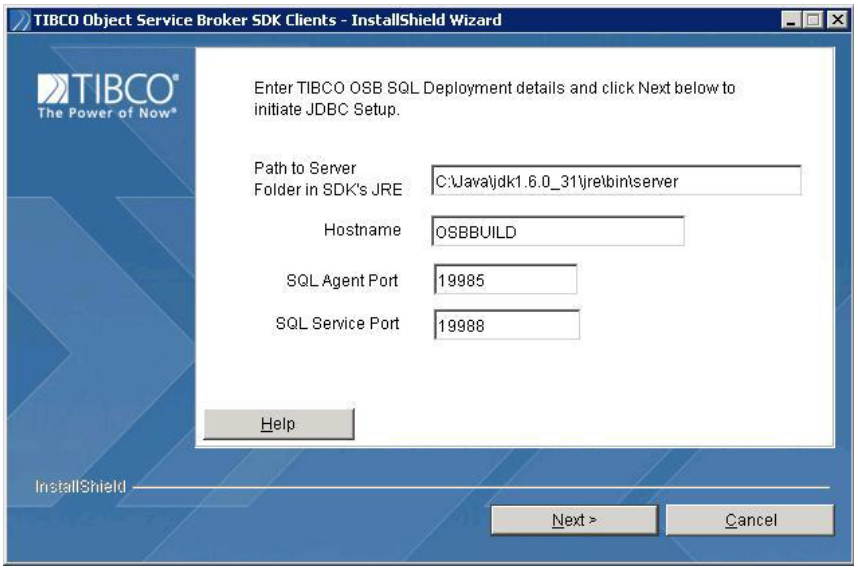
Otherwise, click the **Install** button to proceed with the installation of the SDK Clients.

10. If the following dialog displays, click Yes.

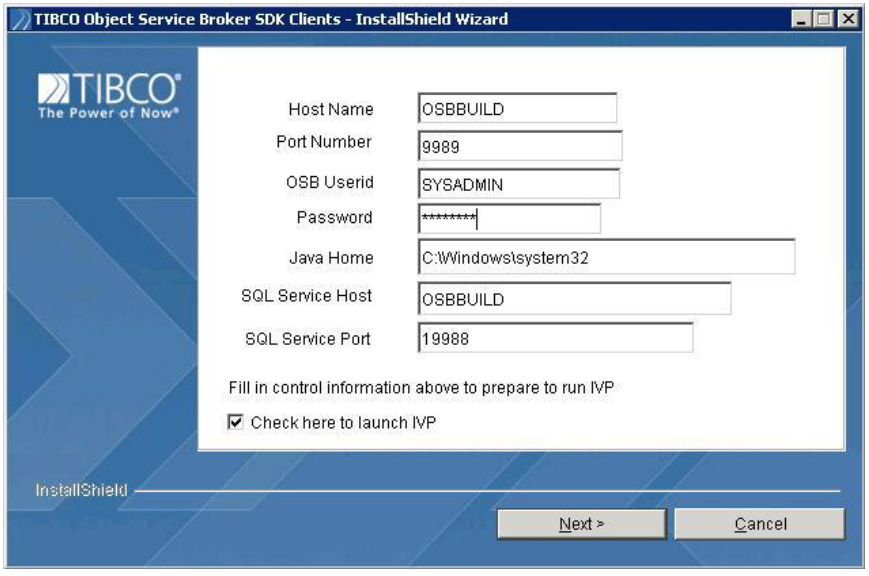


11. The TIBCO Object Service Broker SDK Clients screen appears. The SDK Client installer installs the JDBC SQL Service Adapter, which requires the

parameters shown here. For more information about these parameters, see JDBC Deployment Parameters on page 6.



12. When the SDK Clients installation finishes, the following panel displays:



To run the IVP, specify the field values (described in Installation Verification Procedure Parameters), select the checkbox to launch the IVP, and click the **Next** button.

If this a base installation, a black MS-DOS window for osMon will display and stay active throughout the IVP processing. An hourglass cursor is shown during this process, which can take approximately one minute.

13. When the SDK Clients/IVP installation is complete, click the **Finish** button to exit the installers for those processes.
14. Click the **Finish** button again to exit the installation wizard.

SDK Client Installation Only

If you are only installing the SDK Clients, note the following:

- Use the following installation file:

  ```
  TIB_osbC_6.0.0_win_x86_64.exe
  ```

  The installer requests an installation folder. This folder can be any folder except that in which a pre-Release 6.0 instance of the TIBCO Object Service Broker base components is installed.

- **IVP** A clients-only SDK installation IVP requires that a separate, independent TIBCO Object Service Broker be running to provide the 'host connectivity' that is exercised by the verification process.

Post Installation

After installing TIBCO Object Service Broker, you should perform the following tasks:

Verify Permission Requirements for Data Object Broker Users

As a result of improvements in security in Windows 7 and Windows 2008 Server, the user that starts the Data Object Broker must have the user right "Create Global Objects". By default, only Windows system administrators have this user right. If you choose not to use a system administrator account to start the Data Object Broker, you must use the Security Configuration Manager to modify this local security policy to grant the "Create Global Objects" right to the user that starts the Data Object Broker.

Configure the System

Configure the Data Object Broker and user sessions. If you use distributed data access, set up your communications environment. These procedures are described in subsequent chapters.

Configure an ODBC Data Source (optional)

For Windows installations that include the TIBCO Object Service Broker Adapter for JDBC-ODBC, configure an ODBC data source. For details, see *TIBCO Object Service Broker for Open Systems External Environments*.

Test the Installation

To verify that TIBCO Object Service Broker is correctly installed, even after the IVP has completed successfully, you should start a session:

- To establish a background process for your database, start the Data Object Broker. For details, see Starting the Data Object Broker on page 76.
- To establish a background process for your application development, start an Execution Environment. For details, see Starting osMon on page 96.
- To start a development session, start ostty. For details, see Starting ostty on page 99. The TIBCO Object Service Broker workbench starts with a user of SYSADMIN.

Uninstalling TIBCO Object Service Broker

If another product is dependent on the product you wish to uninstall, you are informed that you must uninstall the other product first.

Use one of the following to uninstall TIBCO Object Service Broker:

- Click **Start > Programs > TIBCO > TIBCO Object Service Broker x.x > OSB >Uninstall**
- Use Add/Remove Programs from the Control Panel.
- Navigate to the `_uninst` directory located in the TIBCO Object Service Broker home directory and invoke the `uninstall.exe` program.

The uninstallation process will explicitly exclude from removal the DATABASE directories and some binary files that are shared between an independent SDK Client and a base installation.

The installation process creates two subdirectories, `_uninst` and `_uninstc`, for the uninstallation components of the OSB Base and SDK Clients. You might be able to start the uninstallation process by invoking the corresponding `uninstaller.exe` program.

Installing on Solaris Systems

Before starting the installation procedure, ensure that your system meets the hardware and software requirements, and that you have reviewed the pre-installation steps.

Installation Files

Installation file names for TIBCO Object Service Broker vary by version number and platform, using the following general format:

`TIB_osb_`*`version_platform`*

where *version* is the three-digit version number for this TIBCO Object Service Broker release and*platform* is an abbreviated form of the hardware platform for which the executable is intended.

The installation files on Solaris for this release are the following:

```
TIB_osb_6.0.0_solaris.bin       (Base product)
TIB_osbC_6.0.0_solaris.bin      (SDK Clients)
```

Download the product from the TIBCO Software web site by following these steps:

1. Contact TIBCO Software for a password, directory information, and so on.
2. Connect to the TIBCO Software web site with the required information.
3. Download the installation file appropriate for your platform.

Choosing an Installation Mode

Table 5 describes the modes for installing the product on Solaris.

Table 5 Installation Modes on Solaris

Installation Mode	Description

GUI mode	Allows you to input values in panels. Use the following in a terminal window:
	`cd installation-directory`
	`TIB_osb_6.0.0_solaris.bin`

Table 5 Installation Modes on Solaris

Installation Mode	Description

Console mode	Allows you to install the software without using a GUI. The installer will prompt you for values. Use the following in a terminal window:
	`cd installation-directory`
	`./TIB_osb_6.0.0_solaris.bin -console`

Silent mode	To use this mode, you must first generate a response file (using GUI mode or Console mode) that contains the input values you want to use for the installation. See Install and Generate a Response File below for details.
	The silent installation ivocation varies based on whether you are performing an OSB and clients combined installation, or are installing clients only.
	• Perform these steps if the JDBC Deployment parameters to be used in the OSB and SDK Clients combined silent installation have non-default values:
	1. Create a text file, *JDBC-DetailsFileName*, containing the JDBC values to be used for the silent JDBC Deployment. The text file must use the format:
	`-V V_DlgIptJRE="`*path*`"`
	`-V V_DlgIptHost="`*name-of-host*`"`
	`-V V_DlgIptAgentPort="19985"`
	`-V V_DlgIptDataSvcPort="19988"`
	`-V V_DlgIptJDBCcheckBox="true"`
	where *name-of-host* is the name of the machine where the service will run and *path* is the location of the JDK as shown here:
	On Windows: `"`*JDK_HOME*`/jre/bin/server"`
	On Solaris: `"`*JDK_HOME*`/jre/lib/sparc/server"`
	On Linux: `"`*JDK_HOME*`/jre/lib/i386/server"`
	JDK_HOME is the full root path to the Java Development Kit installation.
	2. Run the silent installation using the invocation:
	`./TIB_osb_6.0.0_solaris.bin -silent -options` *responseFileName* `-V V_JDBC_DETAILS=`*JDBC-DetailsFileName*

Table 5 Installation Modes on Solaris

Installation Mode	Description

	• Perform these steps if you are installing SDK Clients only and the response file and JDBC Deployment parameters were generated using console mode:
	1. Add the following line to the response file:
	`-V V_JAVA_STARTER_DIR="`*JDK_HOME*`"`
	The addition must immediately follow this line in the response file:
	`-P installLocation=....`
	2. Type the following at the command prompt to run the silent installation:
	`cd installation-directory`
	`./TIB_osb_6.0.0_solaris.bin -silent -options` *responseFileName*

Install and Generate a Response File	You can generate a response file during installation which you can later use to invoke the installer with the selected values as default values (GUI mode) or as selected values (silent mode). To install and generate a response file, type the following at the command prompt: `./TIB_osb_6.0.0_solaris.bin -options-record` *responseFileName*
GUI or Console mode Install Using a Response File	You can use a previously generated response file for installation. For non-silent modes, the response file determine the defaults that are presented. For silent mode, the response file determines what will be installed. To install using a response file, type the following at the command prompt: `./TIB_osb_6.0.0_solaris.bin -options` *responseFileName* (GUI mode) `./TIB_osb_6.0.0_solaris.bin -console -options` *responseFileName* (Console mode)
Combining Options	You can combine the different available options. For example, to install using Console mode and generate a response file, use: `./TIB_osb_6.0.0_solaris.bin -console -options-record` <*responseFileName*>

Installing the Software

The following procedure is for a new installation. For instructions on upgrading from a previous release, see the section titled "Migration from Previous Releases on Open Systems" in *TIBCO Object Service Broker Release Notes*.

Before proceeding, you should have determined which components you will install as described in Base Components and SDK Clients on page 2. If you will install the SDK Clients, ensure that the SDK Clients installer file has been placed in the same folder as the base components installer, and that this folder is writable (*before* invoking the base product installer). Approximately 5 KB of available space is needed.

Base Components Installation (with or without SDK Clients installation)

When using GUI mode, the Windows and Solaris installers use similar steps and panels. For examples of these panels for a Windows installation, see Installing the Software on page 10.

To install TIBCO Object Service Broker, perform the following:

1. Initiate installation by issuing the command of the desired installation mode.
2. When prompted, accept the license agreement.
3. When prompted, select New installation.
4. Specify the installation directory where the product will be installed. The default installation directory depends on who performs the installation:
   - For root users, the default installation directory is `/opt/tibco`.
   - For non-root users, the default installation directory is `/<myhome>/tibco`, where <*myhome*> is the home directory of the user.
5. Select the components to install. If you wish to install SDK Clients, specify that the SDK Clients installer should be launched.
6. Specify the database engine parameters (described in Installation Parameters).
7. Specify the system parameters (described in Installation Parameters).
8. When a summary of your installation selections displays, click the **Install** button.
9. When the base component installation finishes, the SDK Clients installer is automatically launched. If you did not choose to install SDK Clients, go to

Step 13. Otherwise, click the **Install** button to proceed with the installation of the SDK Clients.

10. If a dialog regarding osbinstall.data message displays, click Yes.
11. When the SDK Clients installation finishes, the IVP installer is automatically launched. To run the IVP, specify the field values (described in Installation Verification Procedure Parameters), select the checkbox to launch the IVP, and click the **Next** button.

    If this a base installation, a black window for osMon will display and stay active throughout the IVP processing.
12. When the SDK Clients/IVP installation is complete, click the **Finish** button to exit the installers for those processes.
13. Click the **Finish** button again to exit the installation wizard.

SDK Client Installation Only

If you are only installing the SDK Clients, note the following:

- Use the following installation file:

  `TIB_osbC_6.0.0_solaris.bin`

  The installer will request an installation folder. This folder can be any folder except those where the TIBCO Object Service Broker base components have already been installed.
- The installation command by mode are as follows:

  GUI mode – `TIB_osbC_6.0.0_solaris.bin`

  Console mode – `TIB_osbC_6.0.0_solaris.bin -console`

  Silent mode – `TIB_osbC_6.0.0_solaris.bin -silent -options <responseFileName>`

Setting Environment Variables

After installing TIBCO Object Service Broker, you must run a utility (`osbenv`) for setting up environment variables.

installdir`/utils/osbenv`

The utility must be run with ". " (period, blank) preceding the command (you can also add this to your .profile file).

Post Installation

After installing TIBCO Object Service Broker, you should perform the tasks described in this section.

Configure System Parameters

Solaris system parameters are in the /etc/system file. Consider adjusting the following parameters in this file to allow for a very large number of TIBCO Object Service Broker users on one system, to provide the semaphores required for each user:

- SEMMNS — the maximum number of semaphores on the system — should be set to its default value plus the value of the MAXUSERS Data Object Broker parameter.
- SEMMSL — the maximum number of semaphores per set — should be set to the larger of its default value or the value of the MAXUSERS parameter.
- RLIM_FD_MAX — the maximum number of file descriptors for the system — must be increased to allow for the required number of users plus a contingency amount, because a file handle is used for each user logging in.

See *TIBCO Object Service Broker Parameters* for information about the MAXUSERS Data Object Broker parameter.

Configure Cron Queue b for the SCHEDULE Statement

Use of the SCHEDULE statement for asynchronous processing results in at-jobs being submitted for immediate execution to queue b under the control of cron. You might need to update the characteristics for cron queue b in the /etc/cron.d/queuedefs file to increase the number of at-jobs that might run simultaneously. Set the maximum number of simultaneous jobs for queue b to its default value **and** the anticipated system-wide number of at-jobs submitted with the SCHEDULE statement that must execute concurrently.

For details, see the manual pages for cron(1M), at(1), and queuedefs(4).

Configure the DOB

Configure the Data Object Broker and user sessions. If you use distributed data access, set up your communications environment. These procedures are described in subsequent chapters.

Test the Installation

To verify that TIBCO Object Service Broker is correctly installed, you should start a session:

- To establish a background process for your database, start the Data Object Broker. For details, see Starting the Data Object Broker on page 76.
- To establish a background process for your application development, start an Execution Environment. For details, see Starting osMon on page 96.
- To start a development session, start ostty. For details, see Starting ostty on page 99. The TIBCO Object Service Broker workbench starts with a user of SYSADMIN.

Uninstalling TIBCO Object Service Broker

If another product is dependent on the product you wish to uninstall, you are informed that you must uninstall the other product first.

Navigate to the `_uninst` directory located in the TIBCO Object Service Broker home directory and invoke the `uninstaller.bin` program.

The installation process creates two subdirectories, `_uninst` and `_uninstc`, for the uninstallation components of the OSB Base and SDK Clients. If you install SDK Clients along with the OSB Base, first run `uninstaller.bin` from the `_uninstc` subdirectory. Uninstallation for the OSB Base does not proceed if you do not adhere to that sequence.

Log Files and Environment Variables

Installation Log Files

Install and uninstall log files are created in the `<installdir>\logs` directory.

Environment Variables

On Windows, environment variables are created by the installer at installation time. On Solaris, a utility must be run to create environment variables after the installation is complete. For details, see Setting Environment Variables on page 23.

Table 6 Environment Variables

Variable	Purpose
HURON	Directory used for TIBCO Object Service Broker installation and referenced by the Data Object Broker. Value: `<installdir>`
HURONDIR	Full path and filename of the Data Object Broker directory file. Value: `<installdir>/database/huron.dir`
OS_ROOT	Directory used for TIBCO Object Service Broker installation and referenced by the Execution Environment and TIBCO Object Service Broker monitor process (osMon). Value: `<installdir>`
PATH	The following directory is added at the beginning of the appropriate *PATH* environment variable: Value: `<installdir>/bin;%PATH%`
LD_LIBRARY_PATH (Solaris only)	The following directory is added at the beginning of the *LD_LIBRARY_PATH* environment variable: Value: `<installdir>/sharedlib$LD_LIBRARY_PATH`

Installing Multiple Instances

You can install multiple instances of TIBCO Object Service Broker on the Windows or Solaris platforms by performing the following:

1. Run the installer as described in Installing on Microsoft Windows on page 8 or Installing on Solaris Systems on page 19. You must run the installer separately for each desired instance.
2. Specify a unique directory for the installation of each instance. For example, you could specify the following on Solaris:
   - `/tibco/osb1` – Directory name for the first instance
   - `/tibco/osb2` – Directory name for the second instance

   For a Windows-only installation, make the above values unique by suffixing an instance identifier (IID), a numeric value that is incremented by one for each completed installation: `DOB Nodename`, `DOB IPCKEY`, `MON NAME`, and `SHAREDMEMADDR`. The IID also identifies the Start Menu shortcuts.
3. Specify unique values for the installation parameters:
   - DOB Nodename
   - DOB Alias
   - DOB Port
   - DOB IPCKEY (Solaris only)
   - osMon Port
   - TN3270 PORT
   - osMon IPCKEY (Solaris only)

   For details, see Installation Parameters on page 5.
4. Complete the installation for each instance, and perform any required post-installation tasks for each instance. For example, on Solaris, you must run the `osbenv` utility for setting up environment variables.

Administration

On both Solaris and Windows, there are different ways that you can administer multiple instances of TIBCO Object Service Broker on the same system. You must set the correct environment in order to administer each instance of TIBCO Object Service Broker. This section contains examples of a few simple approaches.

Solaris

This example uses a script to perform the following:

- Sets the correct environment for each TIBCO Object Service Broker instance.
- Starts the Data Object Broker and Execution Environment for the selected instance.
- Starts an xterm for that instance.

This example includes three instances: *Production*, *Development*, and *Test*.

```
#!/bin/ksh
echo "Please select OSB instance:"
echo
echo "Production  1"
echo "Development 2"
echo "Test        3"
read answer'?Selection:> '
if test $answer = "1"
then
echo "Starting Production OSB"
export HURON=/tibco/osb
export OS_ROOT=/tibco/osb
export HURONDIR=/tibco/osb/database/huron.dir
export PATH=/tibco/osb/bin:/tibco/osb/utils:$PATH
export LD_LIBRARY_PATH=/tibco/osb/sharedlib
hrncr
echo
echo "Starting Production osMon"
osMon >> /dev/null &
xterm -T "Production OSB" -n "PRD" -sb -bg white &
fi
if test $answer = "2"
then
echo "Starting Development OSB"
export HURON=/tibco/osb1
export OS_ROOT=/tibco/osb1
export HURONDIR=/tibco/osb1/database/huron.dir
export PATH=/tibco/osb1/bin:/tibco/osb2/utils:$PATH
export LD_LIBRARY_PATH=/tibco/osb1/sharedlib
hrncr
echo
echo "Starting Development osMon"
osMon >> /dev/null &
xterm -T "Development OSB" -n "DEV" -sb -bg white &
fi
if test $answer = "3"
then
echo "Starting Test OSB"
export HURON=/tibco/osb2
export OS_ROOT=/tibco/osb2
export HURONDIR=/tibco/osb2/database/huron.dir
export PATH=/tibco/osb2/bin:/tibco/osb2/utils:$PATH
```

```
export LD_LIBRARY_PATH=/tibco/osb2/sharedlib
hrncr
echo
echo "Starting Test osMon"
osMon >> /dev/null &
xterm -T "Test OSB" -n "TST" -sb -bg white &
fi
ksh .startosb
```

Windows

You can use individual batch files to control each instance of TIBCO Object Service Broker. Three samples are shown below. The first sets the correct environment and starts the Data Object Broker:

```
set HURON=c:\tibco\osb1
set OS_ROOT=c:\tibco\osb1
set HURONDIR=c:\tibco\osb1\database\huron.dir
set PATH=c:\tibco\osb1\bin;%PATH%
hrncr
```

This batch file starts the Execution Environment:

```
set HURON=c:\tibco\osb1
set OS_ROOT=c:\tibco\osb1
set HURONDIR=c:\tibco\osb1\database\huron.dir
set PATH=c:\tibco\osb1\bin;%PATH%
osmon
```

This batch file starts a command prompt with the correct environment set:

```
set HURON=c:\tibco\osb1
set OS_ROOT=c:\tibco\osb1
set HURONDIR=c:\tibco\osb1\database\huron.dir
set PATH=c:\tibco\osb1\bin;%PATH%
color OC
cmd
```

You can group batch files such as these together in a directory associated with an instance of TIBCO Object Service Broker or place them directly on the Windows Desktop.

The installer creates a menu shortcut `OSB CMD` for each instance of TIBCO Object Service Broker. That shortcut invokes a preconfigured batch file similar to the preceding example and enables you to enter and execute TIBCO Object Service Broker commands against that particular instance.

It is assumed that you execute the TIBCO Object Service Broker commands described in the following sections in either of these two ways:

- From a Command Prompt window created by choosing the `OSB CMD` menu shortcut that corresponds to the instance.
- With the required environment variables that you set up according to the examples.

Installing the TIBCO Object Service Broker UI

Overview

Installation Modes

As with the TIBCO Object Service Broker installation, the installer allows you to run in the following modes:

- GUI mode
- Console mode
- Silent mode

For details, see Installation Modes on page 2:

System Requirements

Microsoft Windows

Ensure that your system meets the following minimum requirements:

- A system that runs a supported version of Microsoft Windows. See the readme file for the supported versions.
- 512 MB of RAM; 1 GB is recommended
- 590 MB of free hard-disk space

Solaris

Ensure that your system meets the following minimum requirements:

- A SPARC machine running the Solaris operating system. See the readme file for the supported versions.
- 490 MB of free hard disk space
- A window environment such as CDE (that is, X Windows) is required to run the installer in GUI mode. It is not required for a console installation.

Preparing for Installation

Prerequisite Software

To run the TIBCO Object Service Broker UI, the following software must be installed:

— JRE 1.5 or greater

— Eclipse SDK

See the readme file for the supported versions.

The installer prompts you to specify the location for Eclipse on your system. The installer will not allow you to complete the installation if a valid Eclipse installation is not found. To run the TIBCO Object Service Broker UI, you must install the supported versions of JRE and Eclipse.

If the TIBCO Runtime Agent is installed on the machine where you are installing the TIBCO Object Service Broker UI, do not install the JRE that is bundled with the product.

Installer Account

- **Microsoft Windows** – You must have administrator privileges for the machine on which TIBCO Object Service Broker is installed. If you do not have administrator privileges, the installer exits. You must then log out of the system and log in as a user with the required privileges, or request your system administrator to assign the privileges to your account.
- **Solaris** – TIBCO Object Service Broker can be installed by a regular (non-root) user and super-user (root). Different users can install the same product at different locations.

Installation Files

The installation files for the TIBCO Object Service Broker UI for this release are the following:

`TIB_osbui_6.0.0_win.zip`	Windows
`TIB_osbui_6.0.0_sol.zip`	Solaris

Installing the Software

Install the software as follows:

- On Windows, run the `TIBCOUniversalInstaller.exe` file.
- On Solaris, run t3he `TIBCOUniversalInstaller-sol-sparc.bin` file.

Apply a mode as described in Installing the Software on page 10 (for Windows) or Installing the Software on page 22 (for Solaris). Bear in mind the following exception in the silent mode:

A `.silent` file is available, which requires customization as described in the embedded directions. After customization, to start a silent installation, run the `.exe` or `.cmd` file with the `-silent` flag. With the customized `.silent` file, you need not create a response file for the installer for the OSB Base or for the installer for the SDK Clients.

After starting the installer and accepting the license agreement, you are prompted to specify the Eclipse installation directory (if not detected by the installer).

If you have already installed a previous version of the OSB UI, be sure to define a new *TIBCO_HOME* for the new installation.

Postinstallation

TIBCO Plug-ins File

During installation, you are prompted to specify your Eclipse installation directory. The installer then automatically creates a TIBCO plug-ins file. To apply another Eclipse installation without rerunning the TIBCO Object Service Broker UI installer, do the following before running that UI:

1. In your Eclipse installation directory (referred to here as *ECLIPSE_HOME*), create a `links` subdirectory (if it does not already exist).
2. In the *ECLIPSE_HOME*`/links` directory created in Step 1, create a text file named `TIBCOplugins.link`.

3. Insert the following line in the `TIBCOplugins.link` file:

   `path=`*TIBCO_HOME*`/components`

   where *TIBCO_HOME* is the top level installation directory for TIBCO products where the TIBCO Object Service Broker UI was installed.

Uninstalling the Software

Microsoft Windows

Use one of the following to uninstall TIBCO Object Service Broker UI:

- Click **Start > Programs > TIBCO > TIBCO Object Service Broker UI > Uninstall**
- Use Add/Remove Programs from the Control Panel.
- Navigate to the following directory:

  TIBCO_HOME\components\eclipse\uninstaller_archives\osb_broker_ui\

  and invoke the `uninstall.exe` program.

Solaris Systems

Navigate to the following directory:

TIBCO_HOME\components\eclipse\uninstaller_archives\osb_broker_ui\

and invoke the `uninstall.bin` program.

Installing the Service Gateway for WMQ

Service Gateway for WMQ is an interface used to access IBM WebSphere MQ message queues from within TIBCO Object Service Broker. It ensures that data is presented in a manner consistent with TIBCO Object Service Broker behavior. You can configure the Data Object Broker and Service Gateway for WMQ to reside on different hosts and/or operating systems (z/OS, Windows or Solaris).

You must install the TIBCO Object Service Broker base component before installing Service Gateway for WMQ.

See Also *TIBCO Object Service Broker for Open Systems External Environments* or *TIBCO Object Service Broker for z/OS External Environments* for more information on using Service Gateway for WMQ.

Distribution Media and Contents

This software is distributed as a zip file containing either the dynamic load library to enable the WMQ gateway functionality within the TIBCO Object Service Broker base component on Windows, or the shared object to enable the same functionality within the TIBCO Object Service Broker base component on Solaris.

The following ZIP files comprise the distribution media:

`TIB_srvcgw-wmq_6.0.0_win.zip`	(Windows)
`TIB_srvcgw-wmq_6.0.0_sol.zip`	(Solaris)

Installation

Windows

Once you have obtained the appropriate zip file for this platform, unzip the embedded dynamic link library `s6b.mqsgw.dll` and move it into the `bin` directory of your TIBCO Object Service Broker base component installation.

Solaris

Once you have obtained the appropriate zip file for this platform, unzip the embedded shared object `s6b.mqsgw.so` and move it into the `sharedlib` directory of your TIBCO Object Service Broker base component installation.

IVP Considerations

Before running the IVP, be sure to have the name of a queue manager and that of a

queue from which you can read and to which you can write. From the workbench or in batch, do the following:

1. Run the rule `@MOMIVP_WRITE(`*queue_manager_name*`,`*queue_name*`)`.

   On completion, this message is displayed:

   `MOMIVP: MSG "IVP TEST MSG" WRITTEN TO QUEUE` *queue_name*

2. Run the rule `@MOMIVP_READ(`*queue_manager_name*`,`*queue_name*`)`.

   On completion, this message is displayed:

   `MOMIVP: MSG "IVP TEST MSG" READ FROM QUEUE` *queue_name*

Postinstallation Tasks

Perform the postinstallation tasks as described in this section.

Setting up the Security Framework for Archiving the Audit Log

Be sure to set up the purging of the Audit log data. Otherwise, the segment (usually segment 99), which holds the audit data in the ACCESSLOG table, eventually fills up and processing for TIBCO Object Service Broker stops. That might require a recycling of your Execution Environment.

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

Schedule Monitoring Tasks

Monitor activities with the following facilities:

Facility	Reference
Administration Menu	Chapter 6, Monitoring TIBCO Object Service Broker
TIBCO Hawk	Chapter 7, Using the Interface to TIBCO Hawk and the TIBCO Hawk documentation

The frequency of the tasks depend on your operational requirements. Following are the general recommendations.

Daily Tasks

We recommend the following daily tasks:

- Check the status of the previous night's backup.
- Check the Data Object Broker `hrncr` log for messages, such as the one below, that warn of segment space problems:

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

  You can view the percentage of free pages in each segment with the ADMIN utility option `B SEGMENT/DASD`.
- Check the Data Object Broker `hrncr` log for error message ss, such as the one below, that warn of connection issues:

```
S6BUM014E Unable to connect to node 'ZOSDOBB' or  S6BUM102I
TCP/IP library call failed: connect(), last error=ECONNREFUSED,
Connection refused, fn=hctcConnect, source=hctcpip.c(1174)
```

- Check all Osmon, Execution Environment, and Session logs for any unusual error messages or rule failures. Follow-up on them.

Weekly Tasks

We recommend the following weekly tasks:

- Check for application updates, including those for new applications.
- Check the performance statistics.
- Perform offline segment backups if a window exists and ensure that Batch Pointer Check (`hrnbrptr`) shows that the backups are clean.
- If you are using peer processing or external database gateways, select `ADMIN` option `1 IN-DOUBT TRANSACTIONS` to confirm that no in-doubt transactions are pending. In case of any, find out why TIBCO Object Service Broker has not automatically resolved them or manually fix them yourself.

Bimonthly Or Monthly Tasks

We recommend that you perform these tasks on a bimonthly or monthly basis:

- Verify that the test and certification systems are synchronized with the systems to which you will apply software updates.
- Check for software updates and schedule maintenance at the most opportune time.
- Select the `ADMIN` utility option `A` to review the logical `GET` to physical `READ` processing. Strive for a successful hit ratio of at least 95 percent. Calculate the hit ratio as follows:

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

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

Chapter 2 Configuring the Data Object Broker

This chapter describes how to configure the Data Object Broker.

Topics

Overview

Purpose

This chapter describes how to configure a Data Object Broker, on Windows and Solaris, that can be used by clients and Execution Environments on the same machine or on separate machines and platforms.

Distribution of Components Across Platforms

TIBCO Object Service Broker components that make up one system can be distributed across various platforms, or across different machines on the same platform (for this description, Windows and Solaris are considered the same platform). Here are the components and the platforms where each can run:

Component	Microsoft Windows	Solaris	z/OS
Data Object Broker	Y	Y	Y
Execution Environment	Y	Y	Y
Clients	Y	Y	

Supported Connections

The following connections among these components are supported:

- Connections between components running on the same platform.
- Connections via distributed data. Data Object Brokers on any platform can communicate with each other using a peer-to-peer implementation.
- Connections between a z/OS Gateway and a Data Object Broker on Windows or Solaris. If the Gateway requires a z/OS Execution Environment, you can connect the Execution Environment to the Data Object Broker for this purpose only.
- Connections between an TIBCO Object Service Broker Adapter for JDBC-ODBC on Windows and a z/OS Data Object Broker via an Execution Environment on Windows or Solaris for this purpose only.
- Connections between an ODBC Gateway on Windows and a z/OS Data Object Broker via an Execution Environment running on Windows or Solaris for this purpose only.

- Connections between an Oracle Gateway on Windows or Solaris and a z/OS Data Object Broker via an Execution Environment running on Windows or Solaris for this purpose only.

Chapter 4, TIBCO Object Service Broker Execution Environment and Client Sessions, on page 91 describes how to set up Execution Environments and clients.

We do not recommend or support the use of remote or cross-platform Execution Environments for online use.

Distribution of Data Object Brokers

In a peer-to-peer distributed data system, Data Object Brokers can communicate with other Data Object Brokers on the following platform:

- Via TCP/IP in z/OS, Windows and Solaris

Chapter 5, Configuring a Communication Environment for Access of Distributed Data, on page 105 describes how to set up Data Object Brokers for peer-to-peer communications.

See Also Appendix A, Sample Configurations, on page 171 for the description of a sample TIBCO Object Service Broker configuration, and a series of configuration options.

TIBCO Object Service Broker for z/OS Installing and Operating for details about configuring the Data Object Broker on z/OS.

Processes That Form the Data Object Broker

Starting the Data Object Broker activates TIBCO Object Service Broker. The Data Object Broker operates as a set of processes that work closely together to give the appearance of a single entity. It is started and controlled using the **hrncr** command. The specific processes that are started and the number of processes that are started depend on the Data Object Broker configuration parameters, set in the crparm file in the install_path/database directory.

Refer to TIBCO Object Service Broker Operator Commands on page 85 for more information about the **hrncr** command and to Setting Data Object Broker Parameters on page 46 for more information about the crparm file.

Description of Processes

The processes that form the Data Object Broker are:

hrncr	Supervisor that starts other processes, times events, handles shutdown and abnormal termination, and provides information about the release level of TIBCO Object Service Broker and the state of the Data Object Broker.
hrnappl	Database application that contains TDS and distributed data support. The number of **hrnappl** processes started is determined by the MAXCONCURRENT Data Object Broker parameter.
hrncomwq	Process that acts as an outage notification daemon for IPC communications. This process forwards a disconnection message to the work queue serviced by the **hrnappl** processes on behalf of an abnormally terminating local client.
hrnckpt	Journal manager, which handles checkpoint integrity. A checkpoint is created if the redolog is 50% full or the resident page manager buffers are 15% full.
hrnredo	Redolog and contingency log manager, which handles transaction commit integrity and distributed data integrity. It also handles transaction recovery if the Data Object Broker is started after an abnormal termination or system outage.

hrncomm	Process that handles outbound connections via TCP/IP. An **hrncomm** process is started for every outbound connection made (that is, one process for each thread). Currently, connections are made only to peer Data Object Brokers. If there are no PEERS parameters specified in the Data Object Broker parameter file then no **hrncomm** process is started. Refer to Chapter 5, Configuring a Communication Environment for Access of Distributed Data, on page 105 for more information about communications to peer Data Object Brokers.
hrncomt	Process that handles inbound connections to this Data Object Broker via TCP/IP. The **hrncomt** process is a listening process.

See Also

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for information about checkpoint processing.

TIBCO Object Service Broker Parameters for more information about parameters.

Setting Data Object Broker Parameters

Data Object Broker parameters are initially set at installation. You change these defaults by editing the crparm file. The crparm file is located in the install_path/database directory (install_path is stored in the *HURON* environment variable). You can edit the crparm file by using:

- A text editor, as described in Changing Parameters in the crparm File on page 46
- The TIBCO Object Service Broker Database Administrator tool, as described in The TIBCO Object Service Broker Database Administrator Tool on page 74

You can change some parameter values dynamically by issuing an TIBCO Object Service Broker operator command. For more detail, refer to Dynamically Modifying the Parameters on page 48.

Sample Entries

The following shows a sample set of entries in a crparm file:

```
NODENAME = A
PEERS = (B, 9, 10, WN2)
PEERS = (C, 10, 10, WN3, 2)

MAXDBMS = 0         # Maximum number of DBMS servers (0 to 4096)
MAXUSERS = 32       # Maximum number of seated users (1 to 4096)
MAXOPERATORS = 2    # Maximum number of operators (1 to 64)
```

Changing Parameters in the crparm File

To override the default parameter settings in the crparm file:

1. If the Data Object Broker is running, shut it down.
2. Open the file in a text editor.

   You should copy the line containing the parameter you want to change.
3. Remove the leading comment character (#).
4. Change the value.

Access Control Parameters

The ability to start, stop, and manage a running Data Object Broker is restricted to certain users, specified through the following Data Object Broker parameters: OPERATOR, PRIVILEGED, SYSADMIN.

These users are classified and identified as follows:

Operator	You can specify one or more operators in one or more OPERATOR parameters.
Privileged	You can specify one or more privileged users in one or more PRIVILEGED parameters.
System administrator	You must specify only *one* user to be the TIBCO Object Service Broker system administrator with the SYSADMIN parameter.

All other users are considered general users.

See Also

TIBCO Object Service Broker Managing Security for more information about TIBCO Object Service Broker access control.

TIBCO Object Service Broker Parameters for information on available Data Object Broker parameters and the crparm file.

Dynamically Modifying the Parameters

Data Object Broker parameters are initially set at installation. You can modify some parameters dynamically in two ways:

- Using the **PARM** operator command with **hrncr**
- Using the Operator Command option of the hrntladm (Administration menu) utility

Refer to Setting Data Object Broker Parameters on page 46 and Chapter 6, Monitoring TIBCO Object Service Broker, on page 125 for more information about these methods.

Format of the PARM Operator Command

If you issue the operator command in conjunction with **hrncr**, the format is:

```
hrncr PARM=parmname1=value1,parmname2=value2,parmnameN=valueN
```

If you issue the command using the operator command line, the format is:

```
PARM=parmname1=value1,parmname2=value2,parmnameN=valueN
```

PARM=	The operator command that indicates you want to override a startup parameter.
parmname=	The name of the startup parameter you want to override.
value	The new value for the startup parameter.

Modifiable Parameters

You can modify these Data Object Broker parameters using the PARM command:

CHKTRACE	CLOGRETRYINT	DBPROFILE	ERRNOTRACE
IOTRACE	MSGTRACE	STDERRTRACE	STDOUTTRACE
SVCTRACE	SYNCTRACE	WTOPID	WTOSOURCE
WTOTAG	WTOTIME		

See Also *TIBCO Object Service Broker Parameters* for information on the Data Object Broker parameters and their allowable values.

Data Object Broker Files

Detailed View of the Database Directory

This list shows the structure of the install_path/database directory, created when you install TIBCO Object Service Broker (install_path is stored in the *HURON* environment variable).

Database Configuration Files

dbdef	The Data Object Broker database definition file, containing property definitions for all Data Object Broker special files. For more information, refer to Database Definition File (dbdef) on page 58.
crparm	The Data Object Broker parameter file, defining attributes for your Data Object Broker. Minimal definitions require SYSADMIN and NODENAME. For more information about Data Object Broker parameters, refer to *TIBCO Object Service Broker Parameters*.
huron.dir	Data Object Broker directory file, which can be network-shared and placed anywhere. If it is not shared, it resides in the install_path/database directory. This file is required for the Data Object Broker, and must contain at least the entry of the local Data Object Broker. It lists the Data Object Brokers (nodes) within a distributed peer-to-peer domain. There are several sample entries in the file. For more detail, refer to Data Object Broker Directory File (huron.dir) on page 54.
secparm	Audit log security parameters file. It contains the list of userids that are allowed to run the tools that archive the audit log.

Database Directories

AUDITLOG	Contains the PAGE1 audit log file
CLOG	Contains the CLOG (contingency log) file

JOURNAL	Contains the JOURNAL1 and JOURNAL2 activity log files
METASTOR	Contains the PAGE1, PAGE2, and PAGE3 MetaStor files
REDO	Contains the REDOLOG file
SEG01	Contains the Pagestore files for segment 1 as defined in the dbdef file. Other segments can be created if needed.

For more information about the files in these directories, refer to Data Object Broker Special Files on page 51.

Data Object Broker Special Files

What are the Special Files?

The following files are known collectively as the Data Object Broker special files:

- Three MetaStor files (segment 0)
- One or more segment pages
- Two journals
- Contingency log
- Redolog
- One audit log page

These files are located in the install_path/database directory (install_path is set with the *HURON* environment variable), and have the following default names:

MetaStor	METASTOR/PAGE1
	METASTOR/PAGE2
	METASTOR/PAGE3
Segment 1	SEG01/PAGE1
Journals	JOURNAL/JOURNAL1
	JOURNAL/JOURNAL2
Contingency log	CLOG/CLOG
Redolog	REDO/REDOLOG
Audit log	AUDITLOG/PAGE1

Description and Size of the Special Files

MetaStor

The MetaStor, a symbolic name for segment 0, is the active store for all TIBCO Object Service Broker metadata. It maintains the definitions, characteristics, access paths, and storage locations of all objects in TIBCO Object Service Broker. Every Data Object Broker has its own MetaStor.

The MetaStor must contain at least 27,000 pages (three 9000 4 KB pages, the default). Most of this space is occupied by the unloaded MetaStor image that is part of the TIBCO Object Service Broker distribution. TIBCO Object Service Broker rules, screens, reports, and table definitions that users create are also kept in the MetaStor.

Segment 1

Segment 1 stores TIBCO Object Service Broker users' table occurrences. It should occupy no less than 5,000 pages (the default). For larger TIBCO Object Service Broker systems, segment 1 could occupy tens or hundreds of thousands of pages, and like any other segment, can be divided into two or more page data files. You can have other segments defined (for example, segment 2, segment 3, and others); the default database definition file install_path/database/dbdef defines only the MetaStor, segment 1, and segment 99.

Journals

TIBCO Object Service Broker contains two journals, which perform the following tasks:

- Create write-ahead logs for checkpoint recovery.

  When the system performs a checkpoint, all modified data pages in the resident page manager buffers are copied to the cache portion of the journal, therefore assuring their recoverability in the event of a failure. The resident page manager is an internal Data Object Broker component that manages local memory containing images of all Pagestore pages currently in use.

- Provide an audit trail of all changed physical pages.

  The system switches between journals when one reaches capacity and the contents of the full journal are saved. The procedure to save the full journal is called a *spin*. The journals can be placed on the same SCSI controller as the redolog, but they must be in a different directory.

Contingency Log

The contingency log contains a record of every in-doubt or contingent transaction.

Redolog

The redolog contains records of all update operations recently performed, or about to be performed, against the database. TIBCO Object Service Broker uses the redolog to reconstruct the committed updates made between checkpoints and maintain database integrity.

The redolog file should be defined at the head of a SCSI controller that is a different controller than the one containing the Pagestore page files.

Audit Log

The audit log records access to TIBCO Object Service Broker and its objects.

The full installation process initially places the audit log in the MetaStor (segment 0).

The size of the audit log depends on the level of logging that you set for the Execution Environment via the SECAUDITLOG Execution Environment parameter. This parameter can be set to STRICT, NORMAL, or DISABLED. STRICT causes logging of *all* accesses to objects with the Log Accesses flag set to Y in the Security Manager.

Because of the possible very large size of the log file, the audit log must be located in its own segment.

Use the hrnbrial utility to move the audit log, if you want to move it. The hrnbrial utility does not move the audit log segment if it was already moved once.

See Also

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for more information about journal processing and in-doubt and contingent transactions.

The hrnbrial utility in *TIBCO Object Service Broker for Open Systems Utilities* for information about moving the segment for the audit log.

TIBCO Object Service Broker Managing Security for more information about the audit log facility.

TIBCO Object Service Broker Parameters for more information about the SECAUDITLOG parameter.

Data Object Broker Directory File (huron.dir)

You use the Data Object Broker directory file, huron.dir, to relate Data Object Broker names to the physical machines where they reside by defining the necessary IPC and TCP/IP communication attributes for TIBCO Object Service Broker systems. The Data Object Broker directory file is organized as a series of records that you can modify using a text editor.

To define a TIBCO Object Service Broker system for an Execution Engine, File Gateway, or another TIBCO Object Service Broker system (peer) to connect to it, just add another record to the Data Object Broker directory file. Detailed communication attributes do not have to be remembered when they are added to the Data Object Broker directory file.

Name and Placement of the Directory File

The Data Object Broker directory file path and name defaults to the value of the *HURONDIR* environment variable, which at installation is set to install_path/database/huron.dir. Ensure that the huron.dir file is:

- Network shared
- Accessible to all Data Object Brokers defined in it
- Accessible to all Execution Environment and File Gateway server machines associated with the Data Object Brokers

Also ensure that the environment variable *HURONDIR* is set on all Data Object Broker and Execution Environment and File Gateway server machines.

What the Communication Attributes Define

The attributes in the Data Object Broker directory file define the communication resources needed by each TIBCO Object Service Broker system when:

- Passing messages between TIBCO Object Service Broker components—clients processes, Execution Environments, external database servers—and their assigned Data Object Broker.

  Data Object Broker communications uses System V shared memory and semaphore IPC resources when the components and Data Object Broker are on the same machine. It uses TCP/IP when the components and the Data Object Broker are on different machines.

- Optionally, passing messages between peer TIBCO Object Service Broker systems through peer-to-peer distributed data connections using TCP/IP for communications.

Attributes Defined

Each record in the huron.dir file defines a set of communication attributes for one Data Object Broker. These attributes are used by components that must communicate with the Data Object Broker such as client processes, Execution Environments, external database servers, and peer Data Object Brokers:

node	Each record begins with the word *node* followed by a series of keyword attributes.
name	The communications identifier that identifies the Data Object Broker to connect to. If the value is greater than eight characters an alias attribute must be defined in the huron.dir file. On Windows and Solaris, this value is specified by the NODENAME Data Object Broker parameter. On z/OS, this value is specified by the COMMID Data Object Broker parameter.
alias	If *name* exceeds eight characters, an *alias* of up to eight characters must be specified; if the name is eight characters or fewer, *alias* is optional. If specified, the alias is used as the communications identifier for communications among the z/OS, Windows, and Solaris, environments.
host	The machine where the Data Object Broker operates can be specified as either an IPV4 or IPV6 address or as a fully qualified symbolic host name (for example, host=`machine.world.com`). For host names, the system *hosts* file is used to resolve the name to an IP address).
ip	The TCP/IP address of the host. Specified using IPV4 or IPV6 format. This should be used only if the host name is not defined to TCP/IP.
service	The service name used by this node for TCP/IP communications. 1 to 8 alphanumeric characters.

port	A four-digit number greater than 1024 that is unique for each Data Object Broker. The port is used by TCP/IP and must be unique within each machine for all applications using TCP/IP. Using port numbers greater than 10000 usually ensures that Data Object Broker port numbers do not conflict with other applications.
keepalive	This optional value enables TCP/IP keepalive probes which will facilitate the detection of severed, idle connections, as well as possibly preventing the severing of idle connections by firewalls. On platforms that do not support a TCP/IP keepalive feature, the functionality is emulated with periodic out-of-band messages between TIBCO Object Service Broker components. The value is an integer between 1 and 65535, which specifies the interval in seconds between probes on outbound connections to this node.
ipckey	A three-byte hexadecimal number (0x1 through 0xffffff) that identifies shared memory and semaphores (IPC resources) for each Data Object Broker. Each Data Object Broker must have a unique IPC key within each machine. Its value cannot be 0.
timeout	For Windows only. Specify a timeout value in milliseconds. The range of allowed values is 0 to 2,147,483,647 milliseconds. The default value of 0 represents an infinite timeout period. This parameter specifies the amount of time a peer or peer server waits for a message before giving up. With the use of this parameter, communication failures can be detected when communications are lost in a non-controlled way, because of the use of a KEEPALIVE type of functionality. Peers can then be reconnected automatically when the client site comes back up, after the communications network fails, or after the site of a connected client fails, such as during a power outage. Such failures previously required shutting down the server Data Object Broker and restarting it. For TCP/IP, you should use a value of 60000 and adjust the value depending on observed behavior of the system. If the value is too small, peers could constantly be starting and stopping, while a value that is too large can cause delays in sensing an outage and beginning corrective action. No special conditions are raised when a timeout successfully recovers a connection. For TCP/IP, the timeout is a property of the local node. If the local node services an incoming z/OS remote peer, the timeout value should be set to 0 to disable the timeout processing.

Sample huron.dir File

```
node   name=acamar,
       host=acamar.world.com,
       port=8000,
       ipckey=0x8000

node   name=kabul,
       host=crocus.world.com,
       port=8001,
       ipckey=0x8001

node   name=niagara,
       host=niagara.world.com,
       port=8002,
       ipckey=0x8002
```

Database Definition File (dbdef)

The database definition file, dbdef, defines the Pagestore (segments and page files), which is the actual data store, as well as the audit log, the contingency log, the journal, and the redolog. It resides in the TIBCO Object Service Broker \database directory.

TIBCO Object Service Broker uses the dbdef file during initialization. The dbdef file centralizes all database structure information.

For the structure and syntax of the dbdef file, refer to *TIBCO Object Service Broker for Open Systems Utilities*.

This file should *not* be changed without careful consideration of existing TIBCO Object Service Broker files. When data exists in a particular segment, changing a value in the database definition can cause the segment to become unusable or can corrupt data in that segment.

File Naming Conventions

To provide file naming consistency, several rules apply:

- Valid TDS filenames are PAGE1, PAGE2 ... PAGEn.
- TDS page data files belonging to the same segment do not have to reside in the same directory. They can reside on different disks or in different paths, provided each of them resides in the parent directory named by the segment. For example, the following two page data files reside in the same segment (in Windows):

  ```
  f:\SEG01\PAGE1
  g:\SEG01\PAGE2
  ```

- JOURNAL1, JOURNAL2, REDOLOG, AUDITLOG, and CLOG are the default names for corresponding log files. Each file must reside in the parent directory named in the NAME parameter.

  Refer to Data Object Broker Files on page 49 for more information.

How File Location is Determined

The PATH parameter and FILE subcommand provide information about file location to the Data Object Broker. The following rules apply to the DB line:

- Each DB line can be followed by FILE subcommand lines up to the number of files specified in the ACBS parameter.

- The FILE subcommand overrides the segment/log default specified in the DB line.
- The segment/log default specified in the DB line overrides the *HURON* environment variable.

The path to each TIBCO Object Service Broker file is determined by:

- The PATH value in the FILE subcommand line
- The PATH value in the DB line, if the FILE subcommand is not used
- The value of the environment variable *HURON*, if the PATH parameter is not used on either the DB line or the FILE subcommand line

Changing File Location

Files that are formatted for TIBCO Object Service Broker can be relocated by moving or copying them. They can be used again without reformatting as long as you change dbdef to reflect the new file location. Data Object Broker special files are automatically formatted during a Full installation. Additional page or log files must be formatted manually using the hrntlf*xx* utilities. For each of these utilities you must provide the full path name of the file, as defined in the dbdef file.

Use the hrnbrial utility to move the audit log, if you want to move it. The hrnbrial utility does not move the audit log segment if it was already moved once.

See Also

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for more information about journal processing.

TIBCO Object Service Broker for Open Systems Utilities for more information on the formatting utilities and hrnbrial.

TIBCO Object Service Broker Parameters for more information on parameters.

Customize the TIBCO Object Service Broker Unicode Processing

Overview

You can configure the conversion, collation, and case processing of Unicode data in TIBCO Object Service Broker. TIBCO Object Service Broker comes with a set of source configuration files and you can:

- Accept the default configuration data that is part of the TIBCO Object Service Broker system
- Use one or more of the supplied configuration files to override the default values
- Modify one or more of the supplied configuration files to override the default values
- Specify conversions between Unicode and External User Syntaxes using files from http://dev.icu-project.org/cgi-bin/viewcvs.cgi/charset/data/ucm/

These source files may be used to generate binary configuration files which, if present, are read at initialization time replacing the default configuration data. The defaults in the system correspond to the IBM-037 code page. There are no External User Syntaxes defined by default.

There are five types of configuration data, used for:

- Unicode to EBCDIC mapping
- EBCDIC to Unicode mapping
- Unicode case mapping
- Unicode collation sequencing
- Unicode to/from external user syntax mapping

Format of the Data Files

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

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

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

Unicode to EBCDIC Mapping (UniToEbc)

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

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

For example, here is a portion of a file:

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

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

EBCDIC to Unicode Mapping (EbcToUni)

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

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

For example, here is a portion of a file:

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

An EBCDIC character can be mapped only once.

Unicode Case Mapping (UniCase)

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

- A hex value (0000 to FFFF) corresponding to a Unicode character
- A case indicator: either “U” or “u” if the character is uppercase, or “L” or “l” if the character is lowercase
- A hex value, representing the Unicode point for the same character with the opposite case

For example, here is a portion of a file:

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

Unicode Collation (UniColl)

Data lines contain a single significant field: a hex value (0000 to FFFF) representing a Unicode code point. The data lines list the code points in order of their collation. The file must contain 65,536 unique data lines to specify all possible code points.

Unicode to/from External User Syntax Mapping (UniXC01-UniXC16)

The format is described at http://icu.sourceforge.net/userguide/conversion-data.html. Data lines contain three significant fields, separated by white space:

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

Sample Unicode Configuration Files Provided

This is a list of the names of the sample configuration files shipped with TIBCO Object Service Broker (on Solaris, the names are case sensitive). These files appear in your %HURON%/UnicodeConfig directory. The 3- or 4-digit numbers in the filenames refer to the IBM-xxx EBCDIC code page they are based on. Use the files as they are, or modify copies of them to create the desired configuration specification.

Unicode to EBCDIC Mapping	EBCDIC to Unicode Mapping	Unicode Case Mapping	Unicode Collation
		ucasedef	ucoldef
ue037	eu037		ucol037
ue273	eu273		ucol273
ue277	eu277		ucol277

Unicode to EBCDIC Mapping	EBCDIC to Unicode Mapping	Unicode Case Mapping	Unicode Collation
ue278	eu278		ucol278
ue280	eu280		ucol280
ue282	eu282		ucol282
ue284	eu284		ucol284
ue285	eu285		ucol285
ue297	eu297		ucol297
ue500	eu500		ucol500
ue1140	eu1140		ucol1140
ue1141	eu1141		ucol1141
ue1142	eu1142		ucol1142
ue1143	eu1143		ucol1143
ue1144	eu1144		ucol1144
ue1145	eu1145		ucol1145
ue1146	eu1146		ucol1146
ue1147	eu1147		ucol1147
ue1148	eu1148		ucol1148

Creating Binary Unicode Configuration Files

The unigen utility program is used to convert the source configuration files into binary files which will be read at initialization time. You need to run unigen to create binary files for the first 4 file types if the defaults are not suitable for you. You also need to run unigen for type 5 if you wish to define any external user syntaxes. All binary files generated by unigen should be written to the directory specified by the UNICODEDIR parameter or environment variable (see next section). Usage for the unigen executable is as follows:

```
unigen  n  source  target  format  syntax  codepage  fallback
```

where:

n	A value from 1 to 5 indicating the type of file being processed. 1 = Unicode to EBCDIC translation 2 = EBCDIC to Unicode translation 3 = Unicode casing 4 = Unicode collation 5 = Unicode to/from External User Syntax translation
source	The full path name of the configuration source file.
target	The full path name of the generated binary file.
format	0 = assembler output 1 = C output 2 = binary output The remaining parameters are only needed if n=5.
syntax	A number from 1 to 16 identifying the external user syntax.
codepage	The name of the code page (for example IBM-939).
fallback	A value of either True or False (default) according to whether fallback codes are to be used.

Specifying Unicode Configuration

You use the UNICODEDIR Data Object Broker parameter, the UNICODEDIR Execution Environment parameter, and the UNICODEDIR environment variable (for the offline batch utilities) to specify the directory where the Unicode configuration files reside, for example, %HURON%/database/UNICODEDIR.

Specific filenames are used for the configuration files.

These names are case sensitive on Solaris:

- UniToEbc – Unicode to EBCDIC mapping
- EbcToUni – EBCDIC to Unicode mapping
- UniCase – Unicode case mapping
- UniColl – Unicode collation
- UniXCnn – Unicode to external syntax Xcnn mapping, where nn = 01 to 16

If UNICODEDIR is not specified, no configuration files are read and the default configurations, which are part of the TIBCO Object Service Broker application, are used and no external user syntaxes are defined. If UNICODEDIR is specified, the TIBCO Object Service Broker initialization code looks for each of the first four files in the specified directory. If the file is present, TIBCO Object Service Broker uses it to configure its Unicode processing. Otherwise, it uses the default configuration for that file. All files with names matching UniXCnn (where nn may be from 01 to 16) are read and are used to configure up to 16 external user syntaxes XC01 to XC16.

For example, to override only the Unicode to EBCDIC mapping file, do the following:

1. For correct performance, choose the ue*xxx* file, as supplied with TIBCO Object Service Broker, that corresponds to the NLS code page of your system.
2. [Optional] Modify the file as required.
3. Run **`unigen 1`** *source target* **`2`** where *source* and *target* are the path names of the source and target files.
4. Copy the target file to the Unicode configuration directory specified in your UNICODEDIR Execution Environment and Data Object Broker parameters.
5. Rename it to UniToEbc.

To define external user syntaxes XC01 through XC03, do the following:

1. Select the .ucm files corresponding to the code pages you want to define as your external user syntaxes. Assume that you want to use code page IBM-939 for syntax XC01, IBM-939 with fallback codes for syntax XC02, and IBM-933 for syntax XC03. Also assume that directory D:\sourceConfig contains the files ibm939.ucm and ibm933.ucm and that the desired output directory is C:\Unicode.

2. Run the following:

```
unigen 5 D:\sourceConfig\ibm939.ucm C:\Unicode\UniXC01 2 1 IBM-939
unigen 5 D:\sourceConfig\ibm939.ucm C:\Unicode\UniXC02 2 2 IBM-939 true
unigen 5 D:\sourceConfig\ibm933.ucm C:\Unicode\UniXC03 2 3 IBM-933
```

Customize the TIBCO Object Service Broker @SCHEDULEMODEL Table

You can use the @SCHEDULEMODEL table to execute in batch mode with the use of the SCHEDULE statement. You can add instances to the @SCHEDULEMODEL parameterized table after installation.

Execution Environment Parameters Allowed as Variables

TIBCO Object Service Broker substitutes the following variables with the value of the corresponding Execution Environment parameters for the session:

Parameter	Abbreviation	Description
ACTION	AC	The type of invocation for the initial user rule specified in the RULE parameter.
BROWSE	B	Specifies whether the first transaction of the session can update TDS tables and external databases.
CHARSET	C	The default national character set.
DECIMALSEPARATOR	DECSEP	The character to be used as the decimal separator.
DOB	DB	The name of the Data Object Broker to which a session is to connect. Can also affect binding between sessions.
DSBIFTYPE	DSBT	The file type used to process calls to the @READDSN and @WRITEDSN tools and the load/unload to external files tools.
DSDIR	DSD	The path to the directory containing files accessed with the @OPENDSN tool, import and export tables, and load/unload to external files tools.

Parameter	Abbreviation	Description
DSFIELDSEP	DSFS	The field separator for TEXT type import and export tables.
DSIXFTYPE	DSXT	The file type used to process import and export tables.
DSQUOTE	DSQ	The quote character used for TEXT import tables.
EENAME	EE	The name of the Execution Environment to be used to run the session.
EXECLOCALSIZE	LOCALSIZE	The size of the area for rules local variables.
EXECSTACKSIZE	STACKSIZE	The size of the Executor Scope stack.
INSTLIB	I	The name of the library that holds the rules for the installation promoted from the local libraries.
LIBRARY	L	The name of the local library for rules calls.
MSGLOGMAX	MLM	The size of a user session message log in bytes.
PARM	PM	The arguments for the rule that is being submitted for batch processing.
PASSWORD	P	The user's login password to be passed to TIBCO Object Service Broker.
PRINTDATASET	DATASET	The name of a file to which a TIBCO Object Service Broker generated report is written.
PRINTDEST	DEST	The default printer destination for output generated by TIBCO Object Service Broker.

Parameter	Abbreviation	Description
PRINTXWTR	XWTR	The name of an external writer (XWTR) to be used to print output generated by TIBCO Object Service Broker.
PROFILE	PR	Specifies whether the user's profile information stored within TIBCO Object Service Broker is to be merged by the startup rule with Execution Environment parameters at session startup.
RULE	R	The name and arguments of the first application rule to be invoked after login rules processing completes. Does not apply to SDK (C/C++) or SDK (Java) sessions.
SEARCH	SEA	The library search environment for the first rule to be executed.
SESSIONLOGCLEAR	SLC	Specifies whether to clear the session log before a session is started.
SYSLIB	S	The name of the rules library that contains the rules shipped with the TIBCO Object Service Broker product.
TEST	TE	Specifies whether user sessions should be run in test mode.
TRANMAXNUM	TMN	The maximum amount of transaction memory a single transaction can use.
USERID	U	The TIBCO Object Service Broker session user ID.

Parameter	Abbreviation	Description
VARLDELIMITER	VLD	The character to be used as the left delimiter for enclosing substituted variables in @SCHEDULEMODEL.
VARRDELIMITER	VRD	The character to be used as the right delimiter for enclosing substituted variables in @SCHEDULEMODEL.

See Also *TIBCO Object Service Broker Programming in Rules* and the *TIBCO Object Service Broker Application Administion* for information about the @SCHEDULEMODEL table and the SCHEDULE statement.

Data Object Broker Log Files

Default Log File

The default Data Object Broker log file records the following information:

- Significant activities
- Potential or detected problems
- Miscellaneous trace information

The log file is typically created under the *install_path*/log directory. Every time the Data Object Broker starts, a new log file is created. The filename has the format:

`hrncr.999`

where

999 is a sequence number incremented for each new log file.

Formatting Log Entries for the Default Log File

To some degree, the contents and format of the entries in the log can be modified by parameters. The following Data Object Broker parameters affect the format of entries in the log:

WTOPID	Determines if each message is labelled with the process ID of the daemon process from which it originated.
WTOTAG	Determines if each message includes the user ID and transaction ID (where relevant) that the message concerns.
WTOTIME	Determines if each message includes a timestamp.
WTOSOURCE	Determines if source code location information accompanies error detection messages.

See Also *TIBCO Object Service Broker Parameters* for detailed information about these parameters.

Monitoring the Default Data Object Broker Log File

There are two ways to create a window to monitor the log file:

- Windows: From the Start menu, under TIBCO Object Service Broker, click DOB Log View.
- Windows and Solaris: From a command prompt, issue the following command:

  `hrncr log`<Enter>

You see the last few entries in the log file, and any new entries written to the log file appear.

To stop monitoring the Data Object Broker log, close the window.

Sending Log Files to the Windows Event Log

If you are running on Windows, you can log Data Object Broker messages into the Windows event log. You set this via the EVENTLOG parameter to the crparm file. To use this option, complete the following steps:

1. Add an entry for the EVENTLOG Data Object Broker parameter to your crparm file.
2. Set the value for EVENTLOG to ERROR (for error events only) or ALL.

See Also *TIBCO Object Service Broker Parameters* for information about EVENTLOG.

Chapter 3 Operating the Data Object Broker

This chapter describes how to operate the Data Object Broker.

Topics

TIBCO Object Service Broker Administrator Programs

DOB Administrator Program Menu Icons

The Data Object Broker is started and controlled using the **hrncr** command. On the Windows platform, the most common **hrncr** functions are incorporated into icons in the TIBCO Object Service Broker program menu. This is the minimum required functionality to operate and monitor a Data Object Broker:

- DOB Administrator
- DOB Log View
- DOB Shutdown
- DOB Start

See Also TIBCO Object Service Broker Operator Commands on page 85 for more information about **hrncr**.

The TIBCO Object Service Broker Database Administrator Tool

The TIBCO Object Service Broker Database Administrator tool is a standalone application with a graphical user interface (GUI). Its purpose is to facilitate Data Object Broker configuration tasks such as database backup, restore, and formatting, adding a new segment, and resizing an existing one, without replacing the corresponding regular maintenance tasks. It can also help with Data Object Broker parameter settings, and database configuration and size information.

Running the TIBCO Object Service Broker Database Administrator Tool

You start the TIBCO Object Service Broker Database Administrator tool, when the Data Object Broker is offline, by running one of the following script files:

- In Windows: **runDBAdmin.bat**
- In Solaris: **runDBAdmin**

For more information on running the tool, refer to the TIBCO Object Service Broker Database Administrator online help.

Batch Utilities

As well as the program menu items and the hrncr utility, some batch utilities are provided to help administer the product. You can use some of these utilities to:

- Start and display an Administration menu (hrntladm)
- Display to standard out (*stdout*) the resources and servers for a Data Object Broker (rsview)

See Also *TIBCO Object Service Broker for Open Systems Utilities* for information about rsview and other available utilities.

Chapter 6, Monitoring TIBCO Object Service Broker, on page 125 for information about the Administration menu.

Starting the Data Object Broker

Starting the Data Object Broker–Windows

Use one of the following methods to start the Data Object Broker:

- From the Start menu, under TIBCO Object Service Broker, click DOB Start.
- From a command prompt, issue the following command:

  `start hrncr`<Enter>

- To start **`hrncr`** in the background:

  `start /b hrncr`<Enter>

You can also run the Data Object Broker as a service. Refer to Running a Data Object Broker as a Windows Service on page 82 for details.

Starting the Data Object Broker–Solaris

Use one of the following methods to start the Data Object Broker:

- From a command prompt, issue the following command:

  `hrncr`<Enter>

- To start **`hrncr`** in the background:

  `hrncr &`<Enter>

Startup Messages

Messages similar to the following appear when you start a typical TIBCO Object Service Broker system:

```
...

Log file is 'C:\OSTAR\log\hrncr.006'
Starting hrnckpt
Starting hrnredo
Starting hrnappl(0)
Starting hrnappl(1)
Starting hrnappl(2)
Starting hrncomt
Starting hrncomwq
Data Object Broker started
```

Data Object Broker Log Startup Messages

Messages similar to those shown below appear in the Data Object Broker console log:

```
2011/12/22 13:57:22 S6BUS030I Data Object Broker - TIBCO(r) Object Service Broker
Release 6.0.0.0.004
2011/12/22 13:57:22 S6BUL193I Shared memory size is 119510 Kbytes
2011/12/22 13:57:22 S6BUS014I Starting hrnckpt
2011/12/22 13:57:22 S6BUS002I hrnckpt startup completed (pid=5768)
2011/12/22 13:57:22 S6BUS014I Starting hrnredo
2011/12/22 13:57:23 S6BUS002I hrnredo startup completed (pid=2884)
2011/12/22 13:57:23 S6BUS014I Starting hrnappl(0)
2011/12/22 13:57:23 S6BUS002I hrnappl(0) startup completed (pid=548)
2011/12/22 13:57:23 S6BUS014I Starting hrnappl(1)
2011/12/22 13:57:24 S6BUS002I hrnappl(1) startup completed (pid=2084)
2011/12/22 13:57:24 S6BUS014I Starting hrnappl(2)
2011/12/22 13:57:24 S6BUB002I Segment 'METASTOR' now online
2011/12/22 13:57:24 S6BUB002I Segment 'SEG01' now online
2011/12/22 13:57:24 S6BUB002I Segment 'AUDITLOG' now online
2011/12/22 13:57:24 S6BUS002I hrnappl(2) startup completed (pid=828)
2011/12/22 13:57:24 S6BUS014I Starting hrncomt
2011/12/22 13:57:24 S6BUS002I hrncomt startup completed (pid=5580)
2011/12/22 13:57:24 S6BUS014I Starting hrncomwq
2011/12/22 13:57:24 S6BUS002I hrncomwq startup completed (pid=2864)
```

For more information about the Data Object Broker console log, refer to TIBCO Object Service Broker Operator Commands on page 85.

Critical Messages

When TIBCO Object Service Broker encounters a serious problem, it issues to the operator a critical message, for example:

```
S6BTW005E Highest level index updated for table "DJC_TAB1"
```

which indicates that the DJC_TAB1 table is almost full, and logs this message in the @CRITICALMSGS table. Critical messages are those that require action by the system administrator. Using the @CRITICALMSGS table, TIBCO Object Service Broker ensures that the messages are still available after they disappear from the operator's console.

It is the system administrator's responsibility to delete processed messages from the @CRITICALMSGS table. When this table is full, critical messages are issued to the operator console only.

Here is an example of the @CRITICALMSGS table showing the date and time when the messages it contains were issued:

```
BROWSING TABLE    :  @CRITICALMSGS
COMMAND ==>

                                                        SCROLL: P
      KEY        DATE     TIME   MESSAGE
_ ----------- -------- ------    -
_           1 20050527 152108     .
_           2 20050527 152944     .
_           3 20050527 154547     .
```

By browsing the second row, we see the issued message:

```
TABLE TYPE       : TDS
COMMAND ==>
-------------------------------------------------------------------------------

KEY              :           2
DATE             : 20050527
TIME             : 152944
MESSAGE          : S6BTW005E Highest level index updated for table "DJC_TAB1"
                 :
                 :
```

Restarting the Data Object Broker

After a system failure, you should restart TIBCO Object Service Broker as if you had issued a normal shutdown. TIBCO Object Service Broker automatically checks the redolog and determines whether to perform recovery based on the following:

If...	TIBCO Object Service Broker...
The system was shut down normally.	Performs a cold start.
The system was shut down abnormally.	Performs a warm start with recovery.
The system cannot be restarted safely.	Issues messages to help in problem diagnoses.

See Also

TIBCO Object Service Broker for Open Systems Utilities for a complete description of the **`hrncr`** command and its options and sub commands.

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for information about restarting a system after a system failure.

TIBCO Object Service Broker Messages With Identifiers for information about messages issued when the system cannot be restarted safely.

Shutting Down the Data Object Broker

Data Object Broker Shutdown Commands

Use one of the following methods to shut down the Data Object Broker:

- Windows: From the Start menu, under TIBCO Object Service Broker, click DOB Shutdown
- Windows and Solaris: From a command prompt, issue the following:

  `hrncr shutdown`<Enter>

  or

  `hrncr stop`<Enter>
- From the Administration menu (hrntladm) utility, issue the **`shutdown`** or **`stop`** command using the Operator Command option. For more information, refer to Issuing Commands on page 85.

Order of Shutdown

To ensure an orderly shutdown, sessions logged in to the Data Object Broker should be shut down first. If the Data Object Broker is shut down while sessions are still running, the effect is the same as if the userids were cancelled. Peer servers can be shut down automatically by the Data Object Broker or manually from the SERVICE PROVIDER ACTIVITY option on the Administration menu (refer to Chapter 6, Monitoring TIBCO Object Service Broker, on page 125 for more information).

Data Object Broker Log Shutdown Messages

When the shutdown is requested, messages similar to those shown in the following figure are written to the Data Object Broker console log:

```
2011/12/22 14:03:45 S6BUS010I Data Object Broker shutdown requested
2011/12/22 14:03:45 S6BUS022I hrncomwq stopped
2011/12/22 14:03:50 S6BUS022I hrncomt stopped
2011/12/22 14:04:10 S6BUX009W Checkpoint request ignored, no committed transactions
2011/12/22 14:04:10 S6BUB003I Segment 'METASTOR' now offline
2011/12/22 14:04:10 S6BUB003I Segment 'SEG01' now offline
2011/12/22 14:04:10 S6BUB003I Segment 'AUDITLOG' now offline
2011/12/22 14:04:10 S6BUS022I hrnckpt stopped
2011/12/22 14:04:10 S6BUS112I Redolog surplus capacity=25000
2011/12/22 14:04:10 S6BUS022I hrnredo stopped
2011/12/22 14:04:10 S6BUS009I Data Object Broker stopped
```

For more information about the Data Object Broker console log, refer to TIBCO Object Service Broker Operator Commands on page 85.

Running a Data Object Broker as a Windows Service

With these post-installation instructions, a site can run one or more instances of the Data Object Broker under the control of the Windows services manager. A Data Object Broker that is registered as a service can be set to start automatically at system startup.

Installing the Data Object Broker as a Service

To run the Data Object Broker as a Windows service, it must be defined to the Windows services manager through the Registry. To do this, complete the following:

1. Run the following command:

   `hrncrSvc -i` *service_name*

   where *service_name* is a unique name that you assign to the service.

   A message is then displayed, stating that the service is installed.

2. Open the Services applet from the Control Panel.
3. Locate and click the DataObjectBroker entry.
4. Click Startup.
5. Enter a valid Windows user ID and password to run the Data Object Broker.

   Make sure that this user ID is also defined as the SYSADMIN in the Data Object Broker parameter file crparm. For more information on the SYSADMIN Data Object Broker parameter and the crparm file, refer to *TIBCO Object Service Broker Parameters*.

   You must update this whenever your Windows user ID or password changes.

See Also *TIBCO Object Service Broker for Open Systems Utilities* for information about the hrncrSvc (Install Data Object Broker as a Windows Service) utility and the arguments that it takes.

Starting the Service

Start the service in either of these ways:

From the Control Panel

1. Open the Services applet from the Control Panel.

2. Click the Data Object Broker entry.
3. Click Start.

From the Command Line

Run the command **`net start`** *service_name*.

Setting the Service to Start Automatically

1. Open the Services applet from the Administrative Tools menu of the Control Panel.
2. Locate and click the TIBCO Object Service Broker Data Object Broker entry.
3. Click Startup...
4. Set the Startup type to Automatic.

   This causes the Data Object Broker to start when the machine is rebooted. You can shut down and restart your machine if you want to ensure automatic startup is functioning correctly.

Shutting Down Your Windows System

Windows imposes a 20-second time limit for a service to shut down when a general Windows system shutdown is requested. The Data Object Broker cannot shut down in this time and is terminated by Windows.

When the Data Object Broker restarts, it recovers properly. To avoid having Windows terminate the Data Object Broker, stop the service.

Shutting Down a Data Object Broker Started as a Windows Service

There are two ways to shut down a Data Object Broker that started as a Windows service:

- From the Services applet of the Windows Control Panel, select the Data Object Broker service and click on the Close button.
- Run the command **`net stop`** *service_name*.

Each method closes the service properly and advises the Windows service manager of the shutdown.

Uninstalling the Data Object Broker Windows Service

To uninstall the Data Object Broker service, run the following command:

hrncrSvc -u *service_name*

TIBCO Object Service Broker Operator Commands

Issuing Commands

Using hrncr

You can use the **hrncr** command to send TIBCO Object Service Broker operator commands to a running Data Object Broker. You do this by issuing the operator command as an argument to **hrncr**. The Data Object Broker records error messages or results in the log. If you want, you can dedicate a window to monitoring the Data Object Broker log and issue operator commands from other windows as needed. You can use this to see error messages or results as you enter operator commands.

For example:

```
start hrncr dboffline=1<Enter>
```

varies segment 1 offline, and

```
start hrncr canceluser=usr40<Enter>
```

cancels TIBCO Object Service Broker sessions for user ID usr40.

Using hrntladm

The Administration menu (hrntladm) utility provides an Operator Command option (Option O) that shows the local Data Object Broker log and where you can issue operator commands at the same time. You cannot use the Administration menu Operator Command option to view remote Data Object Broker logs.

For more information about the Administration menu, refer to Chapter 6, Monitoring TIBCO Object Service Broker, on page 125.

Available Arguments

-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory, *%HURON%/database*.

-f	Prevent **hrncr** from closing its console window. When, in Windows, hrncr is invoked with a **start** command from a command prompt or is launched from an icon, it creates a temporary console window in which it displays a message for each of the Data Object Broker processes it creates. This argument keeps the window (which can be minimized by the user) open as long as the Data Object Broker is running.
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is extracted from the environment variable *HURON*. If the *HURON* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is the log directory under the TIBCO Object Service Broker installation directory, *%HURON%*/log. This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (*%HURON%*/database/crparm). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory, *%HURON%*/database/crparm.

Available Operator Commands

You can enter the following operator commands using **hrncr** or the Operator Command option as described:

?	Displays command usage information.

CANCELUSER=*sessionid* where *sessionid*=*userid[.terminalid]* The optional *terminalid* may be found in the osMon log, the DOB log or using the hrntladm utility.	Disconnects a session from the Data Object Broker or Execution Environment specified by jobname. Can be issued from a command prompt as an argument to **hrncr**. **Example**: `CANCELUSER=USR001`
CHECKPOINT	Forces an immediate checkpoint. Can be issued from a command prompt as an argument to **hrncr**.
DBOFFLINE=*segname* or *segnumber*	Moves the specified segment offline. Can be issued from a command prompt as an argument to **hrncr**.
DBONLINE=*segname* or *segnumber*	Varies the specified segment online. Can be issued from a command prompt as an argument to **hrncr**. **Example**: `DBONLINE=7`
KILL	Stops the Data Object Broker without the normal shutdown procedure. This sub command should be used only if the Data Object Broker is not responding to a shutdown command or if there is an urgent need to bring it down quickly.
LOG	Creates a window to monitor the Data Object Broker log file. Updates to the log appear as they are written to the log.
NOTRACE=*sessionid* where *sessionid*=*userid[.terminalid]* The optional *terminalid* may be found in the osMon log, the DOB log or using the hrntladm utility.	Terminates tracing of the indicated session.

PARM= *parmname=value*	Overrides the specified Data Object Broker startup parameter. Refer to Dynamically Modifying the Parameters on page 48 additional information about this command. **Example**: `PARM=CHPAGELIMIT=500`
RESUME	Not used
SHUTDOWN	Shuts down the Data Object Broker. Can be issued from a command prompt as an argument to **`hrncr`**. The request is sent to the active Data Object Broker. The shutdown process can take anywhere from half a minute to several minutes depending on Data Object Broker activity shortly before shutdown.
SPINSUBMIT	Submits a journal spin job, either immediately (I) or deferred until the next checkpoint (D). **Example**: `SPINSUBMIT=I`
STATE	Returns the state of the Data Object Broker: Active Online Offline Shutdown
STOP	Shuts down the Data Object Broker (same as the SHUTDOWN command). Can be issued from a command prompt as an argument to **`hrncr`**. The request is sent to the active Data Object Broker. The shutdown process can take anywhere from half a minute to several minutes depending on Data Object Broker activity shortly before shutdown.

STOPSERVER= ALLDBMS | ALLHURON | ALL*server_type* | ALLREMOTE | SRVID*serverid* | *server_userid*	Disconnects one or more external servers. Can be issued from a command prompt as an argument to **hrncr**, where: *server_type* is a three-character abbreviation for the type of external server; for example, SQL based servers have a type SLK, peer servers (also referred to as rules API servers) have type PRS. ALLDBMS stops all external database servers. ALLHURON stops all inbound connections. ALL*server_type* stops all servers of type *server_type*; for example, ALLPRS stops all peer servers. ALLREMOTE combines ALLDBMS and ALLPRS. SRVID*serverid* stops all external servers with ID *serverid*. *server_userid* stops the server with the user ID *server_userid.* **Example**: `STOPSERVER=ALLDBMS`
TRACEID=*sessionid* where *sessionid*= *userid[.terminalid]* The optional *terminalid* may be found in the osMon log, the DOB log or using the hrntladm utility.	Traces all system service activity for the specified session. **Note** You can specify a full system service trace (using the value SSTRACE), or the trace for a specific session.

See Also

TIBCO Object Service Broker for Open Systems Utilities for a complete description of the **hrncr** command and its options and sub commands.

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for information about checkpoint processing.

The appropriate *TIBCO Service Gateway* manual for information about external database servers

Chapter 4

TIBCO Object Service Broker Execution Environment and Client Sessions

This chapter describes how to configure and run TIBCO Object Service Broker client sessions.

Topics

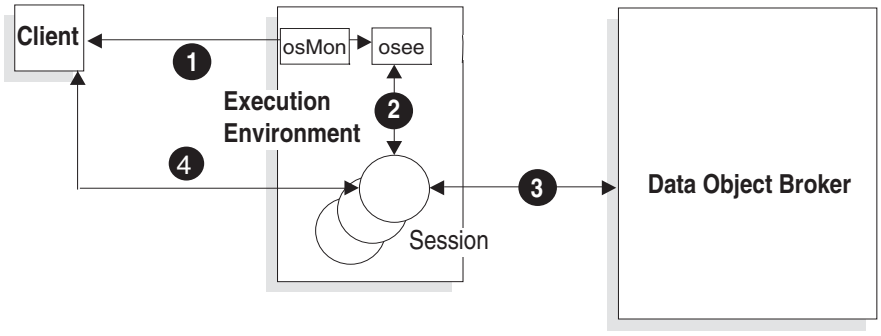
Overview

Purpose

This chapter describes how to configure and start client sessions on Windows and Solaris. These sessions can run on the same machine as the Execution Environment and Data Object Broker or on machines and platforms separate from the Execution Environment and Data Object Broker.

Activity Flow for an ostty User

The following shows the sequence of events when a user logs in from a client:



The numbers in the illustration correspond with the following events:

1. The client selects an Execution Environment and requests that a session be created.

   The osMon process, which an administrator would have started earlier, starts an osee. During initialization, the Execution Environment obtains Execution Environment and session default information.

2. osee activates the session, passing the user's client defaults to the session.

   The session obtains the Data Object Broker name, session defaults, and interfaces for the Execution Environment binding areas. It merges client and session defaults, and opens the session log.

3. Activation is completed by connecting to the Data Object Broker and performing login security processing.
4. The client interacts with the session via screen displays and PF keys pressed.

Actions Initiated by a Client

When a client is started, the following actions take place:

1. An osee starts if one is not already running.
2. A session starts within the Execution Environment.
3. The first rule of the session starts.

   If you are using the TIBCO Object Service Broker UI or the ostty client, you can start up additional sessions within your Execution Environment when the initial session is started.

   When one of these clients is shut down, the following actions take place:
4. The session ends.
5. If no other user is logged in to the Execution Environment, the osee ends.

Communications Between Clients and Servers

TIBCO Object Service Broker uses:

- The IPC mechanism between the Execution Environments and the Data Object Broker when they reside on the same machine
- TCP/IP when they reside on different machines

See Also Appendix A, Sample Configurations, on page 171 to see how these are used in a sample TIBCO Object Service Broker installation.

TIBCO Object Service Broker for Open Systems External Environments for a more detailed activity flow.

Setting Client Session Characteristics

Using Execution Environment Parameters

Execution Environment parameters set the characteristics of an TIBCO Object Service Broker Execution Environment and the default characteristics of its user sessions. The parameters can be set in a number of places, and are evaluated and merged in a pre-defined order before a session begins. Refer to *TIBCO Object Service Broker Parameters* for information about specific Execution Environment parameters, the registry items, and details about the evaluation and use of the parameters.

Operating TIBCO Object Service Broker Clients

Types of TIBCO Object Service Broker Clients

The following table describes the TIBCO Object Service Broker clients:

Client Name	Refer to ...	Description
TIBCO Object Service Broker UI	The TIBCO Object Service Broker UI help file; and *TIBCO Object Service Broker for z/OS Installing and Operating*	Provides a graphical user interface (GUI) to develop applications.
Object Integration Gateway	*TIBCO Object Service Broker Object Integration Gateway* and the Object Integration Gateway help files.	Provides a graphical development interface for creating and modifying TIBCO Object Service Broker and Object Integration Gateway objects.
TIBCO Object Service Broker Adapter for JDBC-ODBC	*TIBCO Object Service Broker for Open Systems External Environments*.	Provides an application program interface into TIBCO Object Service Broker to access data in ODBC-compliant data sources.

Client Name	Refer to ...	Description
osBatch	Starting osBatch on page 101.	Provides a command line interface to start and control the execution of applications.
ostty	Starting and Exiting from ostty on page 99.	Provides a text-only interface to start and control the execution of applications.
SDK (C/C++)	*TIBCO Object Service Broker for Open Systems External Environments*	Provides an application program interface into TIBCO Object Service Broker for use when accessing an Execution Environment on a remote system.
SDK (Java)	*TIBCO Object Service Broker for Open Systems External Environments*	Provides an application program interface into TIBCO Object Service Broker for use when accessing an Execution Environment on a remote system in a Java environment.

Prerequisites

The following prerequisites must be in place before you can activate a client:

- You must define your session parameters in the appropriate place. Refer to *TIBCO Object Service Broker Parameters*.
- The *OS_ROOT* environment variable must be set to the installation folder.
- A Data Object Broker must be available and active.

  Refer to Starting the Data Object Broker on page 76 for details.
- The TIBCO Object Service Broker monitor process (osMon) must be active.

  Refer to Starting osMon on page 96 for details.

Starting osMon

To start osMon, do one of the following:

- Windows: From the Start menu, under TIBCO Object Service Broker, click osMon.
- Windows and Solaris: From a command prompt, issue the following:

  osMon [DOB=*dobname***]**

  You must provide the DOB argument if it does not appear in your mon.prm file.

Starting osMon as a Windows Service

You can set up osMon as a Windows service and, if you want, have it start automatically when you start Windows.

Running osMon as a Windows Service

To run osMon as a Windows service, it must be installed to make it available to the Windows services manager through the Registry. You must run the following command only once:

osMonSvc -i *service_name*

where *service_name i*s a unique name you assign to the service.

You should see a message telling you that the service is installed.

While installing the osMon service, you can specify that it depends on a Data Object Broker service that is running. Run the command—

osMonSvc -i *service_name* **-d** *dob_service_name*

—to inform Windows that the Data Object Broker service *dob_service_name* must be running before the osMon service starts. Afterwards, Windows automatically starts the Data Object Broker service before the osMon service, if the former is not already running.

If the osMon configuration is named something other than the default name `DEFAULT` in the `mon.prm` file, you must specify that name as a parameter override when installing osMonSvc as a service. For example, if the osMon configuration name is `DEFAULT1`, run this command to install the osMon as a service with that configuration:

osMonSvc -i *service_name* **-o name=DEFAULT1**

Feel free to combine the `-o` override option and the `-d` dependency option:

osMonSvc -i *service_name* **-d** *dob_service_name* **-o name=DEFAULT1**

Starting the osMon Windows Service

You can start the osMon service in either of these two ways:

- **From the Control Panel**

1. Open the Services applet from the Control Panel.
2. Click the entry osMonSvc.
3. Click Start.

- **From the command line**

  Run the **`net start`** *service_name* command.

To have osMon start automatically:

1. Open the Services applet from the Control Panel.
2. Locate and click the entry osMonSvc.
3. Click Startup.
4. Set the startup type to Automatic.

   This causes osMon to start when the machine is rebooted. You can shut down and restart your machine if you want to ensure automatic startup is functioning correctly.

Stopping the osMon Windows Service

To stop the osMon service, run the **`net stop`** *service_name* command.

Uninstalling the osMon Windows Service

To uninstall the osMon service, run the **`osMonSvc -u`** *service_name* command.

Getting Help on osMon

To get basic help with osMon commands, run the **`osMon HELP`** command.

Reloading Parameter Settings

To reload the Execution Environment and session parameter settings for a running osMon, run the following command:

`osMon RELOADPARMS [PORT=`*port***`]`**

For an explanation of the variables in the commands, refer to Getting the Variable Values for osMon Commands on page 102

Inquiring into the Status of osMon

Details

To check on the status of osMon, from a command prompt, issue the following:

```
osMon STATUS=DETAIL
```

In response, you find out the following information:

- The Execution Environment name and instance
- The process ID
- The session counter values
- The session instance numbers in the session map for all managed osee processes

Sample output, where two sessions are running:

```
osmon at host=[] port=9068 is: online

name:     [DEFAULT]:0
pid:      141
counters: max=25, total=2 (actual=2, starting=0, pending=0)
map:      0 1
```

Summary

You can also use the following command to get a summary status report, which tells you the number of osee processes and the number of sessions running:

```
osMon STATUS=SUMMARY
```

Sample output:

```
osmon at host=[] port=9068 is: online
osee processes: 1
sessions:       2
```

Starting and Exiting from ostty

Starting ostty

To start ostty, do one of the following:

- Windows: From the Start menu, under TIBCO Object Service Broker, click ostty.

  The menu item is a shortcut to a text interface (*install_path*`\bin\ostty.exe`).

- Windows and Solaris: From a command prompt, issue the following command:

  ```
  ostty
  ```

  The Execution Environment starts automatically and a workbench similar to the following appears.

- Windows: ostty requires the use of physical or virtual function keys. The ostty function keys F1 through F12 are directly supported on a Windows keyboard. ostty function keys F13 through F24 are emulated by pressing Shift+F1 through Shift+F12.
- Solaris: The ostty function keys F1 through F24 are emulated by pressing Ctrl+F, followed by two digits corresponding to the function key number. For example, ostty function key F3 is emulated by pressing Ctrl+f, followed by digit 0 (zero) and digit 3.

Default Developer's Workbench Illustrated

```
HURON            USR40    TEST: N BROWSE: N  2:54 PM    THURSDAY JUL 24  2012

   ER edit rule      ==>                                   SU MO TU WE TH FR SA
   EX execute rule   ==>                                          1  2  3  4  5
   DB debug rule     ==>                                    6  7  8  9 10 11 12
   BR browse table   ==>                                   13 14 15 16 17 18 19
   ED edit table     ==>                                   20 21 22 23 24 25 26
                                                           27 28 29 30 31
   OS object set     ==>
   DS define screen  ==>
   DR define report  ==>
   DT define table   ==>
   DL define library ==>
   GR generate rpt   ==>

   COMMAND ==>  __



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

Exiting from the ostty Client

To exit from the TIBCO Object Service Broker workbench and the ostty client, press PF3 from the workbench.

Starting the TIBCO Object Service Broker UI

To start the TIBCO Object Service Broker UI, perform the following:

1. Start Eclipse.
2. Access the OSB Perspective from the Eclipse menu by selecting **Window> Open Perspective>Other.** The Other Perspective dialog displays.
3. Select **OSB Perspective** and click **OK**.

For information about using the TIBCO Object Service Broker UI, refer to the TIBCO Object Service Broker UI online help.

Starting osBatch

The osBatch utility runs an TIBCO Object Service Broker batch client as a standard Windows or Solaris application. The executable is located in *install_path*/bin. To start osBatch, issue the **osBatch** command, from a command prompt, with the parameters you require, as shown in this example:

```
osBatch DOB=PROD U=JOEB P=XYZ123 R=BATCH1 SEARCH=L NOBROWSE
LIBRARY=PRODBATCH
```

Ending a Session

Other than pressing PF3, you can also end an Execution Environment session by one of the methods shown below (for an explanation of the variables in the commands, refer to Getting the Variable Values for osMon Commands on page 102).

Shutdown of a Session

To end a session, from a command prompt, issue the following:

```
osMon STOPSESS=eename:eeinstance:sessioninstance
HOST=host,PORT=port
```

Shutdown of an Execution Environment

To shutdown an Execution Environment from a command prompt, issue the following:

```
osMon CLOSEEE=eename:eeinstance HOST=host,PORT=port
```

The TIBCO Object Service Broker monitor process shuts down the Execution Environment as soon as all its sessions are terminated.

Immediate Shutdown of an Execution Environment and All Its Sessions

To stop immediately from an Execution Environment and its sessions, from a command prompt, issue the following:

```
osMon STOPEE=eename:eeinstance HOST=host,PORT=port
```

Exiting from osMon

For an explanation of the variables in the commands that follow, refer to Getting the Variable Values for osMon Commands on page 102.

Shutdown

To exit from osMon, from a command prompt, issue the following:

```
osMon CLOSE [HOST=host,PORT=port]
```

The TIBCO Object Service Broker monitor process shuts down its listening port and waits for any child Execution Environment processes to terminate before exiting.

Immediate Shutdown

To exit immediately from osMon and all its child processes, from a command prompt, issue the following:

```
osMon STOP [HOST=host,PORT=port]
```

Immediate Shutdown of All osMon Processes

To exit immediately from all osMon processes running on your machine and all their child processes, from a command prompt, issue the following:

```
osMon STOPALL HOST=host,PORT=port
```

Getting the Variable Values for osMon Commands

To get the values of the variables used in the commands in this section, perform these steps:

1. Open the appropriate log file.

   The log files reside in your \install_path\log folder and have a default name based on the structure explained under the EELOG or the SESSIONLOG Execution Environment parameter, for example, session.DEFAULT.DEFAULT.DEFAULT.12.17-11.07.47.0.0.log.

2. Use the parameter values shown for the command you want to use.

For example, to shut down a session, you would check on the values of the EENAME, EEINSTANCE, and SESSIONINSTANCE parameters in the session log file and issue a command similar to the following:

```
osMon STOPSESS=DEFAULT:0:0
```

Location of parameters:

Parameter	Log File
HOST	mon
PORT	mon
EENAME	session
EEINSTANCE	session
SESSIONINSTANCE	session

See Also

TIBCO Object Service Broker for Open Systems Utilities for details on osBatch.

TIBCO Object Service Broker Getting Started for information on using the developer workbench.

TIBCO Object Service Broker Parameters for information on Execution Environment parameters.

Chapter 5 Configuring a Communication Environment for Access of Distributed Data

This chapter describes how to configure a communication environment to access distributed data.

Topics

Overview

Using Distributed Data

Users can run an application on one TIBCO Object Service Broker system that accesses data on one or more other TIBCO Object Service Broker systems. This feature, known as distributed data, requires the configuration of a communications environment. This environment uses TCP/IP for communications for *all* supported connections, as well as SNA for connections between z/OS systems.

When the communications environment is set up, you must configure and manage peer servers to pass the data between connected TIBCO Object Service Broker systems.

Supported Connections

You can set up distributed data on any supported connection configuration. Refer to Distribution of Components Across Platforms on page 42.

The following sections describe how to set up your communications environment to support each of these distributed-data connections.

See Also Appendix A, Sample Configurations, on page 171 for the description of a sample TIBCO Object Service Broker configuration, and a series of configuration options.

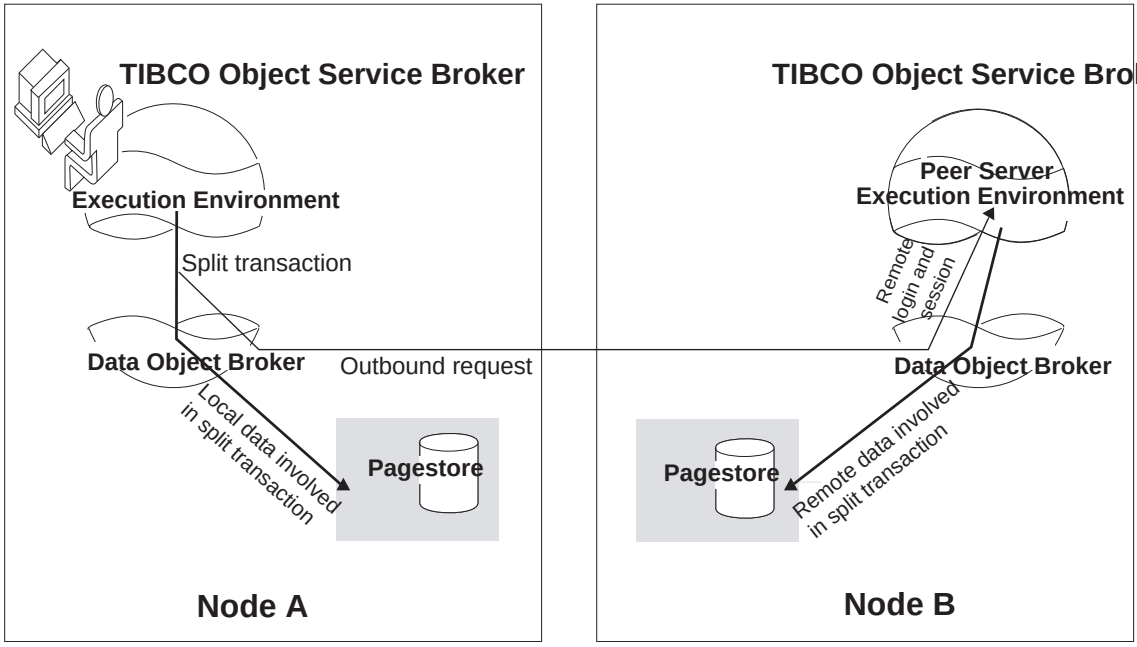
TIBCO Object Service Broker for z/OS Installing and Operating for details about configuring the Data Object Broker on z/OS.

TIBCO Object Service Broker Application Administion for additional information about distributed data implementation for application development.

Managing Peer Servers

You require peer servers within an Execution Environment for peer-to-peer distributed data between TIBCO Object Service Broker nodes (a *node* is equivalent to a Data Object Broker). The peer server provides the distributed data services for a Data Object Broker receiving incoming (or inbound) connections from a peer node. The following sections describe how to set up peer-to-peer distributed data connections between Data Object Brokers across various platforms.

The following graphic shows an outbound request, from Node A's point of view, of data that is distributed on Node B:



Defining a Peer Server

On Windows and Solaris, peer servers are established only if the PEERS Data Object Broker parameter is included in the Data Object Broker parameter file, crparm, located in the install_path/database directory.

On z/OS, you define peer servers through the PEERSERVERID, PEERSERVERNUM, TDS, and MDL Execution Environment parameters and the Resource Management facility available from the Administration menu.

See Also

Connecting Windows to Windows or Solaris on page 111 for details about crparm requirements

Connecting Windows or Solaris to z/OS Using TCP/IP on page 115 for details about the PEERSERVERID and PEERSERVERNUM Execution Environment parameter and Resource Management requirements

Appendix A, Sample Configurations, on page 171 for sample configurations

TIBCO Object Service Broker Parameters for detailed information about the Execution Environment and Data Object Broker parameters

TIBCO Object Service Broker for z/OS Installing and Operating for detailed information about the Resource Management facility

Starting a Peer Server

A peer server starts implicitly when a client session that uses the peer server starts.

Shutting Down a Peer Server with the STOPSERVER Command

You can shut down a peer server using the Data Object Broker operator command **`STOPSERVER`**, for example:

- `hrncr STOPSERVER=SVRIDserverid`
- `hrncr STOPSERVER=server_userid`
- `hrncr STOPSERVER=ALLPRS`

The value for *serverid* or *server_userid* is case sensitive. Refer to Available Operator Commands on page 86 for more information about **`hrncr`**.

Automatic Restart of a Peer Server

If a logical error is detected during peer server operation and the server terminates, the Execution Environment logs the error and restarts the server. When a peer server terminates normally, the server session is not restarted and the total number of peer or rules-based servers is reduced by one.

Format of Peer server_userid

Peer server_userids must be unique to a Data Object Broker, as multiple Execution Environments with peer servers can connect to the Data Object Broker. Peer server_userids are generated in the following format:

```
@EERRRXX
```

where:

`@`	is the at sign, with which all server_userids start, to differentiate servers from regular users.
EE	contains the first two characters of the name for the Execution Environment.
RRR	is the three-character string resulting from a hash on the Execution Environment name.
XX	contains the two alphanumeric characters (base 36) derived from the server number for the server in the Execution Environment. An Execution Environment associates a unique identifier for each peer server it starts. Alphanumeric characters are used rather than numbers to allow for more than 99 servers. Example: `@HR2WU01`

Monitoring Peer Servers

Use the User Activity option in the TIBCO Object Service Broker Administration Utility (option I) to display peer server activity. Press PF2 to view a region list. Select SERVER and press PF2 to display available peer servers. You can also use the rsview utility to view, on standard out (stdout), a report about your peer server connections.

See Also Chapter 6, Monitoring TIBCO Object Service Broker, on page 125 for more information about the User Activity option.

TIBCO Object Service Broker for Open Systems Utilities for details about rsview.

Peer Server Logs

The Data Object Broker log displays information about peer server activity such as startup, termination, and restart. In addition, a peer server log (also called a session log) is generated for each peer server.

Data Object Broker Log

This log displays all peer server activity for a particular Data Object Broker. The following sample messages would appear for one peer server:

```
2012/09/22 14:55:06 S6BUA021I User '@HR2WU01' logged on from 'CC001'
2012/09/22 14:55:13 S6BUA023I PRS server DEFAULT0(THR2WU01) logged on from '@HR2WU01'
```

where:

@HR2WU01	is the peer server user ID. (that is, the session for a peer server). A session log is also generated with the user name as filename.
PRS	is the server type.
DEFAULT0	is the default peer server ID.
THR2W01	is the peer server_userid.

Session Log

The peer server log is written to the path specified by the SERVERLOGPATH Execution Environment parameter.

Connecting Windows to Windows or Solaris

This section describes a sample connection between two TIBCO Object Service Broker Data Object Brokers (*node A* and *node B*) on two Windows machines connected via TCP/IP. The sample would be the same for a Windows to Solaris connection. Node A is the local node and node B is the remote node with peer servers attached. This connection is referred to as an outgoing or *outbound* connection from node A.

Refer to Managing Peer Servers on page 107 for more information about configuring and operating peer-to-peer distributed data connections.

Configuring Node A

1. In the Data Object Broker parameter file (crparm) specify the following parameters:

NODENAME=*nodename*	*nodename* is the Data Object Broker name. In this sample configuration, `NODENAME=A`.
PEERS=(*remote_node*, *outbound#*, *inbound#*, *prefix*, *fslevel*)	*remote_node* is the Data Object Broker name of the node to which this server is connecting. *fslevel* should be 2. The following example corresponds to the sample configuration `PEERS=(B,9,10,NTK,2)`.
MAXUSERS=*nn*	*nn* is the number of users needed to accomodate peer users. Valid values are 1 to 4096. MAXUSERS must be large enough to allow incoming peers to log in. Each incoming peer uses 2 user slots, for example, if the total incoming peers is equal to 10, 20 incoming slots of MAXUSERS should be devoted to incoming peers. In this sample configuration, `MAXUSERS=32`.

2. In the Data Object Broker directory file (huron.dir) create two node entries:

   To simplify the procedure, the same port number is used; the port numbers could differ if there is more than one Data Object Broker on a single machine.

```
node   name=A,
           host=brussels.monarch.com,
                port=7209,
                ipckey=0x7209
     node   name=B,
                host=fornax.monarch.com,
                port=7209,
                ipckey=0x7209
```

   Refer to Attributes Defined on page 55 for details about the huron.dir entries.

Configuring Node B

1. In the Data Object Broker parameter file (crparm) specify the following parameters:

NODENAME=*nodename*	*nodename* is the Data Object Broker name of the node to which this server is connecting. In this sample configuration, `NODENAME=B`.
PEERS=(*remote_node*, *outbound#*, *inbound#*, *prefix*, *fslevel*)	*remote_node* is the Data Object Broker name. *fslevel* should be 2. The following example corresponds to the sample configuration `PEERS=(A,9,10,NTK,2)`.
MAXUSERS=*nn*	*nn* is the number of users needed to accomodate peer users. Valid values are 1 to 4096. MAXUSERS must be large enough to allow incoming peers to log in. Each incoming peer uses 2 user slots, for example, if the total incoming peers is equal to 10, 20 incoming slots of MAXUSERS should be devoted to incoming peers. In this sample configuration, `MAXUSERS=32`.

2. Ensure you have the same entries in your Data Object Broker directory file (huron.dir) as node A.

3. In your mon.prm file, specify the following values to the SERVERS Execution Environment parameter to define the peer servers, in the format SERVERS='numberN sessionnameN', where

numberN	Represents the number of inbound connections on all PEER statements, for example, 10.
sessionnameN	Represents a session defined to the NAME parameter in the session.prm file (refer to step #4.), for example, PEERSERV1. It defaults to DEFAULT0 if left blank. If you specify a value on node A, the @PEERSERVERID shareable tool, which is a system-interpreted session table, *must* contain an entry with that name in the SERVERID field.

4. In the session.prm file, specify the following Execution Environment parameter values:

NAME	The name of the session parameter assignment to be used, for example, PEERSERV1.
SERVERTYPE	The server type: in this case, PRS.
SERVERLOGPATH	The path for the server log, for example, D:\Ostar\log.
EENAME	[Optional] The name of the Execution Environment to be used. If EENAME is specified, you must also specify a corresponding NAME parameter in the ee.prm file.

See Also *TIBCO Object Service Broker Shareable Tools* for details about @PEERSERVERID

TIBCO Object Service Broker Parameters for detailed information about the Execution Environment and Data Object Broker parameters

Establishing the Connection

1. Start the Data Object Broker on nodes A and B.

   On node A you should see the following messages in the Data Object Broker log:

```
2012/06/29 14:04:44 S6BUC013I Connection(0) to 'B' established
2012/06/29 14:04:44 S6BUC013I Connection(1) to 'B' established
```

On node B you should see the following messages in the Data Object Broker log:

```
2012/06/29 13:58:49 S6BUA020I Peer user 'NTA00001' logged on from 'A'
2012/06/29 13:58:51 S6BUA020I Peer user 'NTA00002' logged on from 'A'
```

2. Start the peer server on node B (if not already activated by the Data Object Broker).

```
2012/06/29 14:02:38 S6BUA021I User '@HR2WU01' logged on from 'B'
2012/06/29 14:02:58 S6BUA023I PRS server DEFAULT0(THR2WU01) logged on from '@HR2WU01'
2012/06/29 14:04:15 S6BUA021I User '@HR2WU02' logged on from 'B'
2012/06/29 14:04:29 S6BUA023I PRS server DEFAULT0(THR2WU02) logged on from '@HR2WU02'
```

From node A, you can now access TIBCO Object Service Broker distributed data on node B. You must have identical TIBCO Object Service Broker userids on both nodes with the same security access.

Ending the Connection

1. Shut down the peer servers using the operator command:

   `hrncr stopserver=ALLPRS`

2. Shut down the Data Object Brokers.

For more information on **`hrncr`**, refer to Available Operator Commands on page 86.

See Also *TIBCO Object Service Broker Parameters* for detailed information about the Execution Environment and Data Object Broker parameters.

Connecting Windows or Solaris to z/OS Using TCP/IP

This section describes a sample connection between TIBCO Object Service Broker node A on z/OS and node B on Windows using TCP/IP. The sample would be the same for a Solaris to z/OS connection. Configuration files and parameters can differ according to your site requirements. The following tasks are described:

1. Setting up the z/OS System, page 115
2. Configuring TIBCO Object Service Broker for z/OS, page 116
3. Configuring TIBCO Object Service Broker for Windows, page 118
4. Establishing Outbound Connections, page 120
5. Establishing Inbound Connections, page 121

Refer to Managing Peer Servers on page 107 for more information about configuring and operating peer-to-peer distributed data connections.

Recommendations

We recommend that you minimize the amount of data exchanged between Data Object Brokers in this configuration. For example, structure your applications so that they do not do a full table sweep on the remote node.

Assumptions

This procedure assumes the following:

- Users on z/OS can access data in the Windows Data Object Broker.
- Users on Windows can access data in the z/OS Data Object Broker.
- The Windows node name is not greater than 8 characters.
- Host names are specified rather than IP addresses.

Task A Setting up the z/OS System

Have your TIBCO Object Service Broker administrator update the TIBCO Relay Parameter file used by the z/OS TIBCO Object Service Broker to include the communications identifier for the Windows TIBCO Object Service Broker.

Example Relay Parameter File

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <tcpipparms tcbnum="2" maxtcbsockets="50" />
   <directory>
      <node name="HKXU0003">
         <tcpip host="brussels" port="44444" />
      </node>
      <node name="HOMEDOB">
         <tcpip host="192.160.1.101" port="2000" />
      </node>
   </directory>
</relay>
```

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for detailed information about configuring TCP/IP communications for TIBCO Object Service Broker.

Task B Configuring TIBCO Object Service Broker for z/OS

Have your TIBCO Object Service Broker administrator configure TIBCO Object Service Broker and prepare JCL on the z/OS platform for distributed data. Configure the Data Object Broker on z/OS for peer connection to the Windows Data Object Broker using the Resource Management Facility. To invoke the Resource Manager, choose option 3 on the Administration menu (S6BTLADM utility).

In the Resource Manager, do the following:

1. Use resource type HIN for incoming connections, and resource type HRN for outgoing connections.
2. For HIN connections, define an API if one does not already exist.
3. For each of these components, define a schedule that sets day and time of day limits, if desired for these connections.

Sample Peer Server JCL

You must have JCL for the *peer server* (also referred to as the TIBCO Object Service Broker API server). The following shows a sample used for this configuration:

```
//KXU00SR2 JOB (1),'HKXU0003 PEER SRVR',MSGCLASS=Y,TIME=1440
/*JOBPARM  SYSAFF=*,TIME=1440
/*ROUTE       PRINT HRNPRT1
//*********************************************************************
//* START A PEER SERVER (API) FOR DISTRIBUTED DATA FOR NODE A
//*                  VTAM APPLID - HKXU0003
///*********************************************************************
//REMSRVR   EXEC PGM=S6BDR000,REGION=4096K,TIME=1440
//STEPLIB   DD DSN=OSTAR.R32.LOAD,DISP=SHR
//HRNEXTR   DD DSN=OSTAR.R32.LOAD,DISP=SHR
//HRNOUT    DD SYSOUT=*
//HRNPRNT   DD SYSOUT=*
//HRNIN     DD *
PEERSERVERNUM=4,
MDL=OSB9999,
TDS=HKXU0003
/*
```

Include these Execution Environment parameters in your JCL:

PEERSERVERNUM=4	Specifies the number of peer servers.
MDL=OSB9999	[Optional] If not specified, defaults to OSB9999 where 9999 is the suffix that represents that a four-digit number, starting at 0001, is to be used by the identifiers as they are assigned.
TDS=HKXU0003	The VTAM applid of the z/OS Data Object Broker. Corresponds to the COMMID in the z/OS TCP/IP static configuration data set.
PEERSERVERID=	[Optional] Specifies the ID of the peer server. If not specified, defaults to DEFAULT0.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for information about using the Resource Management facility.

TIBCO Object Service Broker Parameters for detailed information about the Execution Environment and Data Object Broker parameters.

Task C Configuring TIBCO Object Service Broker for Windows

Have your TIBCO Object Service Broker administrator configure TIBCO Object Service Broker on the Windows machine for distributed data. The following assumes you installed TIBCO Object Service Broker on your TCP/IP client machine.

1. In the Data Object Broker parameter file, crparm, specify the following parameters:

NODENAME=*nodename*	*nodename* is the Data Object Broker name. In this sample configuration, `NODENAME=b`.
PEERS=(*remote_node*, *outbound#*, *inbound#*, *prefix*, *fslevel*)	*remote_node* is the z/OS Data Object Broker name. *fslevel* should be 2. The following example corresponds to the sample configuration `PEERS=(A,9,10,NTK,2)`.
MAXUSERS=*nn*	*nn* is the number of users needed to accomodate peer users. Valid values are 1 to 4096. MAXUSERS must be large enough to allow incoming peers to log in. Each incoming peer uses 2 user slots, for example, if the total incoming peers is equal to 10, 20 incoming slots of MAXUSERS should be devoted to incoming peers. In this sample configuration, `MAXUSERS=32`.

2. In the Data Object Broker directory file (huron.dir), create two node entries:

```
# z/OS Node
Node            name=a,
                     alias=brussa
                     host=brussels.sdc.com,
                     port=44444

# Windows Node
Node            name=b,
                     host=bewdley.sdc.com,
                     port=2000,
                     ipckey=0x2000
```

Refer to Attributes Defined on page 55 for details on the huron.dir attributes.

3. In your mon.prm file, specify the following values to the SERVERS Execution Environment parameter to define the peer servers, in the format SERVERS='numberN sessionnameN', where

numberN	Represents the number of inbound connections, for example, 10.
sessionnameN	Represents a session defined to the NAME parameter in the session.prm file (refer to step #4.), for example, PEERSERV1. It defaults to DEFAULT0 if left blank. If you specify a value on node A, the @PEERSERVERID shareable tool, which is a system-interpreted session table, *must* contain an entry with that name in the SERVERID field.

4. In the session.prm file, specify the following Execution Environment parameter values:

NAME	The name of the session parameter assignment to be used, for example, PEERSERV1.
SERVERTYPE	The server type: in this case, PRS.
SERVERLOGPATH	The path for the server log, for example, D:\Ostar\log.
EENAME	[Optional] The name of the Execution Environment to be used. If EENAME is specified, you must also specify a corresponding NAME parameter in the ee.prm file.

See Also *TIBCO Object Service Broker Shareable Tools* for details about @PEERSERVERID.

TIBCO Object Service Broker Parameters for detailed information about the Execution Environment and Data Object Broker parameters.

Task D Establishing Outbound Connections

To enable requests from TIBCO Object Service Broker on Windows to the z/OS system, establish an outbound connection between the TIBCO Object Service Broker node on the Windows machine and the node running under z/OS. Follow these steps:

1. Start your z/OS Data Object Broker and bring up the peer servers.

   Your job log should contain messages similar to the following:

```
10.03.03 JOB26919  S6BKC035L INITIATING PEER CONNECTION TO B       :B
10.03.04 JOB26919  S6BKS050I HURON READY  1998MAR27 10:02
...
10.09.04 JOB26919  S6BKC016L U005F001 LOGGING ON (USER BATCH JOB/NET)
10.09.06 JOB26919  S6BKC018I T005F000 SERVER AVAILABLE,TYPE=API,SERVER ID=DEFAULT0
10.09.06 JOB26919  S6BKC018I T005F000 SERVER AVAILABLE,TYPE=API,SERVER ID=DEFAULT0
10.09.06 JOB26919  S6BKC018I T005F001 SERVER AVAILABLE, TYPE=API, SERVER ID=DEFAULT0
10.09.06 JOB26919  S6BKC016L U005F002 LOGGING ON (USER BATCH JOB/NET)
10.09.07 JOB26919  S6BKC016L U005F003 LOGGING ON (USER BATCH JOB/NET)
10.09.07 JOB26919  S6BKC018I T005F002 SERVER AVAILABLE, TYPE=API, SERVER ID=DEFAULT0
10.09.07 JOB26919  S6BKC018I T005F003 SERVER AVAILABLE, TYPE=API, SERVER ID=DEFAULT0
10.09.09 JOB26919  S6BKC016L @OP00600 LOGGING ON (OPERATOR/NET)
```

2. Start your Windows Data Object Broker and bring up the peer servers.

   View the Data Object Broker log on both sides to ensure the connection is successful. On Windows, you should see messages in the Data Object Broker log similar to the following:

```
2012/06/27 11:05:54 S6BUS014I Starting hrncomm(0)
2012/06/27 11:05:54 S6BUS002I hrncomm(0) startup completed
2012/06/27 11:05:54 S6BUS014I Starting hrncomm(1)
2012/06/27 11:05:55 S6BUS002I hrncomm(1) startup completed
2012/06/27 11:05:55 S6BUS014I Starting hrncomm(2)
2012/06/27 11:05:57 S6BUS002I hrncomm(2) startup completed
2012/06/27 11:05:57 S6BUS014I Starting hrncomm(3)
2012/06/27 11:05:57 S6BUA020I Peer user 'BBB00002' logged on from 'D187945@'
2012/06/27 11:05:58 S6BUS002I hrncomm(3) startup completed
2012/06/27 11:05:58 S6BUS014I Starting hrncomm(4)
2012/06/27 11:05:59 S6BUA020I Peer user 'BBB00003' logged on from 'D187945@'
2012/06/27 11:06:00 S6BUS002I hrncomm(4) startup completed
2012/06/27 11:06:00 S6BUS014I Starting hrncomm(5)
2012/06/27 11:06:00 S6BUC013I Connection(0) to 'A' established
2012/06/27 11:06:00 S6BUA020I Peer user 'BBB00004' logged on from 'D187945@'
2012/06/27 11:06:01 S6BUC013I Connection(1) to 'A' established
2012/06/27 11:06:01 S6BUS002I hrncomm(5) startup completed
2012/06/27 11:06:01 S6BUS014I Starting hrncomm(6)
2012/06/27 11:06:02 S6BUS002I hrncomm(6) startup completed
2012/06/27 11:06:02 S6BUS014I Starting hrncomm(7)
2012/06/27 11:06:02 S6BUA020I Peer user 'BBB00005' logged on from 'D187945@'
```

```
2012/06/27 11:06:02 S6BUC013I Connection(2) to 'A' established
2012/06/27 11:06:03 S6BUC013I Connection(3) to 'A' established
2012/06/27 11:06:04 S6BUS002I hrncomm(7) startup completed
2012/06/27 11:06:04 S6BUS014I Starting hrncomm(8)
2012/06/27 11:06:05 S6BUS002I hrncomm(8) startup completed
2012/06/27 11:06:05 S6BUA020I Peer user 'BBB00006' logged on from 'D187945@'
2012/06/27 11:06:05 S6BUC013I Connection(4) to 'A' established
2012/06/27 11:06:07 S6BUA020I Peer user 'BBB00007' logged on from 'D187945@'
2012/06/27 11:06:07 S6BUC013I Connection(5) to 'A' established
2012/06/27 11:06:08 S6BUC013I Connection(6) to 'A' established
2012/06/27 11:06:09 S6BUA020I Peer user 'BBB00008' logged on from 'D187945@'
2012/06/27 11:06:09 S6BUA020I Peer user 'BBB00009' logged on from 'D187945@'
2012/06/27 11:06:10 S6BUC013I Connection(7) to 'A' established
2012/06/27 11:06:10 S6BUC013I Connection(8) to 'A' established
```

On z/OS, you should see messages in your job log similar to the following:

```
11.08.53 JOB26919  S6BKC035L INITIATING PEER CONNECTION TO B       :B
11.08.53 JOB26919  S6BKC016L NTK00001 LOGGING ON (PEER HURON/TCP)
11.08.55 JOB26919  S6BKC016L NTK00002 LOGGING ON (PEER HURON/TCP)
11.08.56 JOB26919  S6BKC016L NTK00003 LOGGING ON (PEER HURON/TCP)
11.08.56 JOB26919  S6BKC016L NTK00004 LOGGING ON (PEER HURON/TCP)
11.08.59 JOB26919  S6BKC016L NTK00005 LOGGING ON (PEER HURON/TCP)
11.09.00 JOB26919  S6BKC016L NTK00006 LOGGING ON (PEER HURON/TCP)
11.09.01 JOB26919  S6BKC016L NTK00007 LOGGING ON (PEER HURON/TCP)
11.09.03 JOB26919  S6BKC016L NTK00008 LOGGING ON (PEER HURON/TCP)
11.09.04 JOB26919  S6BKC016L NTK00009 LOGGING ON (PEER HURON/TCP)
15.31.15 JOB26919  S6BKC016L @OP01020 LOGGING ON (OPERATOR/NET)
```

At this point, connection is successful.

Task E Establishing Inbound Connections

To enable requests from TIBCO Object Service Broker on z/OS to the Windows Data Object Broker, establish an inbound connection between the TIBCO Object Service Broker node on the Windows machine and the node running under z/OS.

Follow these steps:

1. Start your Windows Data Object Broker and start a session to activate a peer server.

   On Windows, you should see messages in the Data Object Broker log similar to the following:

```
2012/06/27 11:06:13 S6BUA020I Peer user 'BBB00010' logged on from 'D187945@'
2012/06/27 11:06:15 S6BUA020I Peer user 'BBB00011' logged on from 'D187945@'
2012/06/27 15:27:35 S6BUA021I User '@HRRJW04' logged on from 'BEWDLEY'
2012/06/27 15:27:38 S6BUA021I User '@HRRJW07' logged on from 'BEWDLEY'
2012/06/27 15:27:39 S6BUA021I User '@HRRJW06' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW02' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW05' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW08' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW03' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW01' logged on from 'BEWDLEY'
2012/06/27 15:27:40 S6BUA021I User '@HRRJW09' logged on from 'BEWDLEY'
2012/06/27 15:27:55 S6BUA023I PRS server DEFAULT0(THRRJW01) logged on from '@HRRJW01'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW05) logged on from '@HRRJW05'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW09) logged on from '@HRRJW09'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW03) logged on from '@HRRJW03'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW07) logged on from '@HRRJW07'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW06) logged on from '@HRRJW06'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW04) logged on from '@HRRJW04'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW02) logged on from '@HRRJW02'
2012/06/27 15:27:56 S6BUA023I PRS server DEFAULT0(THRRJW08) logged on from '@HRRJW08'
```

2. Start your z/OS Data Object Broker.

   View the Data Object Broker log on both sides to ensure the connection is successful. On Windows, you should see messages similar to the ones in step #1. On z/OS, you should see messages similar to the following:

```
11.08.53 JOB26919  S6BKC035L INITIATING PEER CONNECTION TO B       :B
11.08.53 JOB26919  S6BKC016L NTK00001 LOGGING ON (PEER HURON/TCP)
11.08.55 JOB26919  S6BKC016L NTK00002 LOGGING ON (PEER HURON/TCP)
11.08.56 JOB26919  S6BKC016L NTK00003 LOGGING ON (PEER HURON/TCP)
11.08.56 JOB26919  S6BKC016L NTK00004 LOGGING ON (PEER HURON/TCP)
11.08.59 JOB26919  S6BKC016L NTK00005 LOGGING ON (PEER HURON/TCP)
11.09.00 JOB26919  S6BKC016L NTK00006 LOGGING ON (PEER HURON/TCP)
11.09.01 JOB26919  S6BKC016L NTK00007 LOGGING ON (PEER HURON/TCP)
11.09.03 JOB26919  S6BKC016L NTK00008 LOGGING ON (PEER HURON/TCP)
11.09.04 JOB26919  S6BKC016L NTK00009 LOGGING ON (PEER HURON/TCP)
15.31.15 JOB26919  S6BKC016L @OP01020 LOGGING ON (OPERATOR/NET)
```

Ending the Windows Connection

1. Shut down the peer servers using this operator command:

   ```
   hrncr stopserver=ALLPRS
   ```

2. Shut down the Data Object Broker.

For more information on hrncr commands, refer to TIBCO Object Service Broker Operator Commands on page 85.

Ending the z/OS Connection

1. Shut down the peer servers.
2. Shut down the Data Object Broker.

Chapter 6 Monitoring TIBCO Object Service Broker

This chapter describes how to use the Administration Menu to monitor the TIBCO Object Service Broker system.

Topics

Overview

What is the Administration Menu?

The hrntladm (Administration menu) utility is a menu-based facility used to monitor and control your TIBCO Object Service Broker environment. The options available are a subset of the ones available in OSB for z/OS. Some of the options of the Administration menu are also available using the HURON_STATS tool.

The z/OS version of this tool is called S6BTLADM rather than hrntladm.

The TIBCO Object Service Broker Database Administrator tool is available for you to manage your environment when the Data Object Broker is offline. This tool is described in TIBCO Object Service Broker Administrator Programs on page 74.

Displaying the Administration Menu

When your Data Object Broker is running, use one of these methods:

- Windows only: From the Start menu, under TIBCO Object Service Broker, click DOB Admin.
- Windows and Solaris: From a command prompt, issue the command **`hrntladm`**.

- Windows: hrntladm requires the use of physical or virtual function keys. The hrntladm function keys F1 through F12 are directly supported on a Windows keyboard. hrntladm function keys F13 through F24 are emulated by pressing Shift+F1 through Shift+F12.
- Solaris: The hrntladm function keys F1 through F24 are emulated by pressing Ctrl+F, followed by two digits corresponding to the function key number. For example, hrntladm function key F3 is emulated by pressing Ctrl+f, followed by digit 0 (zero) and digit 3.

Administration Menu

```
HRNTLAD1 OSTARSRV              ADMINISTRATION MENU          2012AUG09 15:19:28

  STATISTICS                                  DIAGNOSTIC DISPLAYS

  A  GENERAL                                  U  HURON PAGE IMAGE
  B  SEGMENT/DASD                             X  IN-DOUBT TRANSACTIONS
  F  MESSAGE LENGTH PROFILE
  H  MESSAGE TURNAROUND TIMES
  I  USER ACTIVITY
  L  NODE NAME LIST
                                              LOCAL DIAGNOSTICS

                                              O  OPERATOR COMMAND


                        ENTER SELECTION: ___________


   STANDARD FUNCTIONS:  PF1-HELP  PF3-MENU  PF6-SCREEN PRINT  PF12-EXIT
```

Header Line

The header line shows the following information about the displayed screen:

HRNTLAD1	Screen name.
OSTARSRV	Node name. The Administration menu supports distributed processing. The node name indicates which TIBCO Object Service Broker is being administered. Use Option L to select a different node name.
HURON ADMINISTRATION MENU	Screen title.

The Administration menu normally runs on the same host as the Data Object Broker. If you are on a machine that is remote from the Data Object Broker, have a crparm file on your machine that defines the NODENAME Data Object Broker parameter as the name of the machine where the Data Object Broker is running. Ensure that your machine has a database directory and the appropriate environment variables set. Refer to Chapter 2, Configuring the Data Object Broker, on page 41 for more information about Data Object Broker configuration.

See Also *TIBCO Object Service Broker Parameters* about the NODENAME Data Object Broker parameter.

Categories of Administration Options

The Administration menu divides the menu options into these general categories:

Statistics — Options for displaying statistical information about the TIBCO Object Service Broker system

Diagnostic Displays — Options for monitoring system activity and diagnosing potential problems

Local Diagnostics — The Operator Command (O) displays the local Data Object Broker log file and lets you issue operator commands.

Menu Options and Levels of Security Access

The following table lists the options available from the Administration menu and the security access required by a user for each option. Some options are accessible by all levels of users. For the secure functions on the Administration menu, users have their security accesses categorized in the crparm file under the following Data Object Broker parameters:

- SYSADMIN (allows only one login ID)
- OPERATOR (list of login IDs)
- PRIVILEGED (list of login IDs)

The MAXOPERATORS parameter, which defaults to 2, specifies the maximum number of simultaneous users of the Administration menu.

Category	Option	Level of Security Access	See page
Statistics	Option A – General	User	130
	Option B – Segment/DASD	User	134
	Option F – Message Length Profile	User	141
	Option H – Message Turnaround Times	User	142
	Option I – User Activity	Privileged User	143
	Option L – Node Name List	Privileged User	151
Diagnostic Displays	Option U – Huron Page Image	System Administrator	153
	Option X – In-doubt Transactions	System Administrator	155
Local Diagnostics	Option O – Local Diagnostics: Operator Command	Operator	159

See Also *TIBCO Object Service Broker Parameters* for more information about parameters.

Option A – General

Option A (General) displays statistics accumulated since the system last started.

```
HRNADMA1   OSTARSRV              GENERAL STATISTICS              2012JUL09 15:55:38

   MSG TRAFFIC          DESTINATION          LOGICAL             PHYSICAL
SEND          10901   SUPV             80   GET          30805   READ            608
RECV          10902   COMM              0   PUT            193   WRITE            15
                      CHPT              1   GET4K        11171   REDO            124
LOGON            10   FILE              0   FRE4K           12   JRNL              0
LOGOFF            9   APPL              0   DEFC             0   INDBT             0
                                           STEAL            0
MAXUSR           20                        CTAB         10858   CTAB             86
MAXCON            1   PH0               0   RULE          1696   RULE            152
USERS             0   PH1           10794   SYNC           192
                      PH2              96   GETF            13
POOL              1   SERVER            0   PUTF            12    RES PAGE MANAGER
GETBUF        36397   OPER             12                        FREBFQ        11800
                                           LOCK             0   AGINGQ            0
                                           UNLOCK           0   LRU2Q           189
SNAP              0                                            CHKPTQ            1


                          ENTER-REFRESH  PF2-DELTA DISPLAY
```

General Statistics Screen Fields

The fields on this screen are divided into the following groups:

MSG TRAFFIC	Shows the type of work being done
DESTINATION	Shows the major processing components doing the work
LOGICAL	Shows the processing that occurred using memory resident data
PHYSICAL	Shows the number of physical I/Os or re-acquisitions employed to complete the work being done
RES PAGE MANAGER	Shows information about Resident Page Manager (RPM) components.

In general, the basis for maintaining a cache is to minimize the amount of physical I/O activity when accessing data pages. When RPM receives a page request, there are two possibilities: the page exists in the cache, or it does not. If the page exists, RPM returns it to the requestor. If it does not, a buffer in the cache is reserved and a physical I/O is initiated to read the page into the buffer.

RPM handles the caching of data pages using the LRU-K page replacement algorithm, which is based on the page replacement policy that, in most database systems, is known as “least recently used” (LRU). LRU-K overcomes the limitations of the traditional LRU algorithm by maintaining the history of the last K references to a page. The K in LRU-K is a parameter where K>=2 (K=1 is valid but ignored because it is the same as the traditional LRU algorithm). RPM uses a value of 2 for K. LRU-2 gives improved performance over LRU-1 and is more adaptive to changes in access patterns than LRU-K for K greater than 2.

MSG TRAFFIC

SEND	Number of messages sent to Execution Environments.
RECV	Number of messages received from Execution Environments.
LOGON	Number of logins processed.
LOGOFF	Number of logouts processed.
MAXUSR	The maximum number of online and batch user sessions allowed at one time, as specified by the MAXUSERS Data Object Broker parameter.
MAXCON	The high water mark for all connected sessions, including online and batch user sessions, operator, and external and peer server sessions.
USERS	The current number of connected user online and batch sessions.
POOL	Number of buffer pools.
GETBUF	Number of obtain buffer requests.
SNAP	Number of snap dumps processed.

DESTINATION

SUPV	Number of User Supervisor requests (rollbacks and release locks).
COMM	Number of messages bound for a remote peer.
CHPT	Number of messages sent to the hrnckpt process.
FILE	Number of messages received by all the hrnfile processes.
APPL	Number of times hrnappl was suspended awaiting file I/O.
PH0	Number of transactions recovered at start-up.
PH1	Number of query transactions.
PH2	Number of commit transactions.
SERVER	Number of external server messages.
OPER	Number of operator messages.

LOGICAL

GET	Number of logical page read requests.
PUT	Number of logical page write requests.
GET4K	Number of internal get buffer requests.
FRE4K	Number of internal release buffer requests.
DEFC	Number of deferred CTABLE build requests.
STEAL	Pages copied for reuse during checkpoint processing.
CTAB	Number of control table requests where CTABLE was found in memory.
RULE	Number of rule page requests.
SYNC	Commit requests serviced.
GETF	Number of get free page requests.

PUTF	Number of release free page requests.
LOCK	Number of lock requests.
UNLOCK	Number of unlock requests.

PHYSICAL

READ	Number of pages in.
WRITE	Number of pages out.
REDO	Number of redolog I/O requests.
JRNL	Number of journal file I/O requests.
INDBT	Number of uncommitted transactions held in-doubt.
CTAB	Number of control table requests where the CTABLE was built.
RULE	Number of rule page reads.

RES PAGE MANAGER

FREBFQ	Number of free pages in the Resident Page Manager.
AGINGQ	Number of pages in the aging (correlation) priority queue.
LRU2Q	Number of pages in the LRU2 priority queue.
CHKPTQ	Number of pages in the checkpoint and checkpoint pending queues.

Option B – Segment/DASD

Option B (Segment/DASD) displays segment and disk statistics. The following screens display this information:

- Segment Statistics
- DASD Statistics
- DASD Stats by Page Type
- Change Segment Status

Segment Statistics Screen

The Segment Statistics screen displays all segments known to the Data Object Broker as defined in the *install_path*/database/dbdef file.

```
HRNADMB1   OSTARSRV              SEGMENT STATISTICS            2012AUG09 15:57:04

SEG NAME      TYPE STATUS    JRN     PAGES      USED      FREE      READS     WRITES
  0 METASTOR   TDS ONLINE     Y      27000     23803      3197        611         16
  1 SEG01      TDS ONLINE     Y       5000       299      4701          1          0
 99 AUDITLOG   TDS ONLINE     Y       1200        53      1147          1          0

     ENTER-REFRESH   PF2-DASD STATISTICS   PF7-BACK   PF8-FORWARD   PF11-MODIFY
```

Information Displayed

Column	Function
SEG	Segment number.
NAME	Segment name.
TYPE	Type of data stored.
STATUS	Online or offline.
JRN	Whether (YES or NO) journaling is active.
PAGES	Total number of pages in all data sets within the segment.
USED	Number of pages in use.
FREE	Number of pages that are currently free. This value is in limited output seven-digit format.
READS	Current number of physical reads.
WRITES	Current number of physical writes.

Key Commands

PF1	Access help information about the current screen.
PF2	Display the DASD Statistics screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF7	Scroll backward.
PF8	Scroll forward.
PF11	Display the Change Segment Status screen.
PF12	Exit from the Administration menu.

DASD Statistics Screen

The DASD Statistics screen displays statistical data for each page data file within a given segment.

```
HRNADMB2   OSTARSRV     DASD STATISTICS FOR SEGMENT 0:METASTOR   2012AUG09 15:57:19

DS#   VOLUME    PAGES     USED     FREE        READ     WRITTEN     MAXC    ERRORS
  1   PAGE1      9000     8078      922         225           5        0         0
  2   PAGE2      9000     7834     1166         179           5        0         0
  3   PAGE3      9000     7891     1109         207           5        0         0

     ENTER-REFRESH   PF2-BY PAGE TYPE   PF4-SEG STATUS   PF7-BACK   PF8-FORWARD
```

DASD Statistics Fields

Column	Function
DS#	The relative page file numbers (the actual filenames are PAGE*ds#*).
VOLUME	The name of the page file.
PAGES	The total page capacity of the page file. If the number of free pages is less than 15 percent of the total, the free value is highlighted.
USED	Number of used pages in the file.
FREE	Number of free pages in the file. If the number of free pages is less than 15 percent of the total, its value is highlighted.
READ	The current number of physical reads.

Column	Function
WRITTEN	The current number of physical writes as of the last checkpoint.
MAXC	Not used.
ERRORS	Not used.

Key Commands

PF1	Access help information about the current screen.
PF2	Display the DASD Statistics by Page Type screen.
PF3	Return to the main menu.
PF4	Redisplay the SEGMENT STATUS screen with refreshed data.
PF6	Print the current screen.
PF7	Scroll backward.
PF8	Scroll forward.
PF12	Exit the Administration menu.

The DASD Statistics screen uses the same data as the DASD Statistics By Page Type screen. The data is refreshed only if you press Enter to refresh or PF3 and B to redisplay the Segment Status screen.

DASD Stats by Page Type Screen

The Pagestore contains several page types, each with a specific purpose. The DASD Stats By Page Type screen shows physical reads and writes for three page types: data, index, and other.

```
HRNADMB3  OSTARSRV  DASD STATS BY PAGE TYPE, SEG 0:METASTOR   2012AUG09 15:57:44

             D A T A              I N D E X              O T H E R
DS#        READ     WRITTEN        READ     WRITTEN        READ     WRITTEN
  1           0           3           0           0         225           2
  2           0           2           0           0         179           3
  3           0           1           0           1         207           3

   ENTER-REFRESH  PF2-DASD STATS  PF4-SEGMENT STATUS  PF7-BACK  PF8-FORWARD
```

Page Types Displayed

DATA	These pages contain the actual data rows.
INDEX	These pages contain index structures used to navigate indexes and data pages below them. There are a number of index types, including primary data index, secondary index, and parameterized table index.
OTHER	These pages are a catchall for pages not used for the above purposes. They are primarily internal control pages.

Key Commands

PF1	Access help information about the current screen.
PF2	Display the DASD Statistics screen.
PF3	Return to the main menu.
PF4	Redisplay the Segment Status screen.
PF6	Print the current screen.
PF7	Scroll screen backward.
PF8	Scroll screen forward.
PF12	Exit the Administration menu.

The DASD Stats By Page Type screen uses the same data as the DASD Statistics screen. The data is refreshed only if you press PF4 to redisplay the Segment Status screen.

Change Segment Status Screen

The Change Segment Status screen allows authorized users (operators) to modify TIBCO Object Service Broker segments.

```
HRNADMB4  OSTARSRV              CHANGE SEGMENT STATUS          2012AUG09 14:26:37

                        SEGMENT 0:METASTOR    TYPE=TDS

                        STATUS    CURRENT   REQUESTED

                                  ONLINE



                    PF9  SCHEDULE SEGMENT OFFLINE REQUEST
                    PF10 SCHEDULE SEGMENT ONLINE REQUEST



                PF4-SEGMENT STATUS  PF11-ISSUE STATUS CHANGE REQUEST
```

Displaying the Change Segment Status Screen

If you have authority to control segments, tab your cursor to the desired segment number and press PF11 to display the Change Segment Status screen.

Key Commands

PF1	Access help information about the current screen.
PF3	Return to the main menu.
PF4	Return to the Segment Status screen without issuing a change request.
PF6	Print the current screen.
PF9	Schedule segment offline request; the segment status must be online.
PF10	Schedule segment online request.
PF11	Request that the status change be sent to the Data Object Brokers.
PF12	Exit the Administration menu.

Option F – Message Length Profile

Option F (Message Length Profile) displays the numbers of messages received and sent in different size ranges. LEN is the length of the message in bytes.

```
HRNADM#1  OSTARSRV  MESSAGE LENGTH PROFILE                    2012AUG09 14:37:38
                                                              DELTA TIME     0
 LEN                  PERCENTAGE OF TOTAL                     COUNT       DELTA
RECEIVES ----1----2----3----4----5----6----7----8----9----+
     64 |************************************              |    8171        8171
    128 |****                                              |     801         801
    256 |*                                                 |     266         266
    512 |**                                                |     525         525
     1K |***                                               |     716         716
     2K |*                                                 |     200         200
     4K |*                                                 |     228         228
    >4K |                                                  |       0           0
   SENDS ----1----2----3----4----5----6----7----8----9----+
     64 |                                                  |     101         101
    128 |*                                                 |     245         245
    256 |************************************************  |    9952        9952
    512 |***                                               |     586         586
     1K |                                                  |      16          16
     2K |                                                  |       0           0
     4K |                                                  |       0           0
    >4K |                                                  |       0           0
                                                    SENDS    10907        10907
      ENTER-REFRESH                               RECEIVES   10908        10908
```

Key Commands

PF1	Access help information about the current screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF12	Exit the Administration menu.

Option H – Message Turnaround Times

Option H (Message Turnaround Times) displays the time spent processing queries and commits in the Data Object Broker. A bar graph shows the percentage of total transactions that fall into the specified time range.

```
HRNADM#1  OSTARSRV  MESSAGE TURNAROUND TIME PROFILE          2012AUG09 14:38:01
                                                             DELTA TIME     0
TIME                PERCENTAGE OF TOTAL MESSAGES             COUNT       DELTA
 QUERIES ----1----2----3----4----5----6----7----8----9----+
       2 |*******************************************    |   10022       10022
       4 |                                               |       0           0
       8 |                                               |       0           0
      16 |***                                            |     606         606
      32 |*                                              |     124         124
      64 |                                               |      32          32
     128 |                                               |       8           8
     >>> |                                               |       2           2
 COMMITS ----1----2----3----4----5----6----7----8----9----+
       2 |*******************************               |      61          61
       4 |                                               |       0           0
       8 |                                               |       0           0
      16 |*************                                  |      25          25
      32 |****                                           |       7           7
      64 |*                                              |       2           2
     128 |*                                              |       1           1
     >>> |                                               |       0           0
                                                 QUERIES     10794
   ENTER-REFRESH                                 COMMITS        96
```

Key Commands

PF1	Access help information about the current screen.
PF3	Return to the main menu.
PF6	Print the current screen.
PF12	Exit the Administration menu.

Option I – User Activity

Option I (User Activity) displays active connections to the Data Object Broker and provides information about those connections. The following screens display this information:

- Active User List
- Activity Detail
- Active Sessions
- Region Selection List
- Connections in Region
- Logical Locks

Active User List Screen

The Active User List screen displays active connections to the Data Object Broker.

```
HRNADMI1  OSTARSRV              ACTIVE USER LIST              2012AUG09 16:39:39

    U-USR20          U-USR50          U-USR51          U-USR30          U-USR52
    U-USR55          U-USR00          U-USR80          U-USR05          U-USR40
    U-USR41

                ENTER-REFRESH  PF2-ACTIVITY DETAIL  PF4-REGION LIST
```

Key Commands

PF1	Access help information about the current screen.
PF2	Place the cursor on desired ID and press PF2 to display the Activity Detail screen.
PF3	Return to the main menu.
PF4	Display the Region List screen.
PF6	Print the current screen.
PF12	Exit the Administration menu.

Activity Detail Screen

The Activity Detail screen is divided into four sections:

1. Session identification (type, terminal, and communication)
2. Current message, activity snapshot
3. Accumulated statistics, queries, commits, and others
4. External resource allocation by stream and type

```
HRNADMI2  OSTARSRV          ACTIVITY DETAIL FOR USR40           2012AUG09 12:59:13

 TYPE OPERATOR         IDLE TIME 00000:12 EST. CPU         0    CROSS-MEMORY N
 TERMINAL TTYP0        REGION OPERATOR    JOB NAME 269          STEP N/A
 TRACE N

 CURRENT MESSAGE IDLE-NO MESSAGE                EXCEPTION
 SEGMENT        PAGE            TABLE

 QUERIES         0                          GETS           0
 COMMITS         0                          PUTS           0
 SERVERS         0                          READS          0
                                            RULES          0

 STR  TRX ID    SERVERS
  01  00000008



 ENTER-REFRESH PF2-LOCKS PF4-USERS PF5-REGIONS PF10-TOGGLE TRACE PF11-CANCEL
```

Session Identification

TYPE	The type of region under which the session is connected, for example, Workstation, Operator, TSO, CICS, Native Execution Environment, and others.
IDLE TIME	The number of minutes and seconds since the last action was requested.
EST. CPU	Estimated CPU time accumulated since the connection was initiated. This does not include some of the shared overhead costs and should not be used for accounting purposes.
CROSS-MEMORY	OSB for z/OS only.
TERMINAL	The terminal used to connect to TIBCO Object Service Broker if the ID was supplied at connection time.
REGION	The REGION or group code to which the session is attributed.
JOB NAME	The process ID of the Execution Environment.
STEP	OSB for z/OS only.
TRACE	OSB for z/OS only.

Current Message

CURRENT MESSAGE	The type of request in progress, for example, query or commit.
EXCEPTION	If the request is waiting on an TIBCO Object Service Broker resource, an exception is identified to indicate which resource.
SEGMENT, PAGE, and TABLE	If a request is in progress, these three values identify the current segment and page numbers and the table name last referenced by the request.

Accumulated Statistics

QUERIES	The number of QUERIES since the connection was made.
COMMITS	The number of COMMITS since the connection was made.
SERVERS	The number of requests to SERVERS since the connection was made.
GETS	The number of GETS since the connection was made.
PUTS	The number of PUTS since the connection was made.
READS	The number of READS since the connection was made.
RULES	The number of RULES executed since the connection was made.

External Resource Allocations

STR	Stream number.
TRX ID	Logical lock identifier.
SERVERS	Type and server connection ID of external resources.

Key Commands

PF1	Access Help information about the current screen.
PF2	Display the Logical Locks screen.
PF3	Return to the main menu.
PF4	Return to the Active Users screen.
PF5	Display the Region Selection List screen.
PF6	Print the current screen.
PF7	Scroll forward.
PF8	Scroll backward.

PF10	Change the trace status of a user session (if authorized).
PF11	Issues a cancel user request (if authorized).
PF12	Exit the Administration menu.

Active Sessions Screen

The Active Session screen lists environment data for each connection of a given user ID. It is available for batch regions where one user ID can be logged in several times, and enables you to locate a particular connection.

```
HRNADMI4  OSTARSRV          ACTIVE SESSIONS FOR USR41          2012AUG09 13:04:17

     TYPE               REGION    TERMINAL  JOB NAME  STEP
     OPERATOR           OPERATOR  TTYP0     257
     OPERATOR           OPERATOR  TTYP0     269

                ENTER-REFRESH  PF2-SELECT SESSION  PF4-USER LIST
```

Active Sessions Fields

The Active Sessions screen displays the following information:

TYPE	The type of region under which the sessions are connected.
REGION	The group name set at connection time.
TERMINAL	The terminal ID through which the session is connected.

JOB NAME	The process ID of the Execution Environment.
STEP	OSB for z/OS only.

Region Selection List Screen

The Region Selection List screen displays region codes within your TIBCO Object Service Broker environment. A region is a grouping code that helps organize users into groups. If a region code is not supplied at connect time, it is set based on the connection types.

```
HRNADMI5  OSTARSRV            REGION SELECTION LIST            2012AUG09 13:06:13

OPERATOR          WORK STN




                ENTER-REFRESH  PF2-SELECT REGION  PF4-USER LIST
```

Key Commands

PF1	Access Help information about the current screen.
PF2	Place the cursor beside the desired region and press PF2 to display a list of connections within that group.
PF3	Return to the main menu.
PF4	Redisplay the Active User List screen, if any user is connected.
PF6	Print the current screen.
PF7	Scroll forward.
PF8	Scroll backward.
PF12	Exit the Administration menu.

Connections in Region Screen

The Connections in Region screen lists all connections with the requested region code. The following sample screen shows connections for the region group code OPERATOR:

```
HRNADMI1  OSTARSRV       CONNECTIONS IN REGION OPERATOR       2012AUG09 13:09:31

O-USR40
```

```
ENTER-REFRESH  PF2-ACTIVITY DETAIL  PF4-REGION LIST
```

Key Commands

PF1	Access Help information about the current screen.
PF2	Display the activity detail for the user.
PF3	Return to the main menu.
PF4	Display the Region Selection List screen.
PF6	Print the current screen.
PF7	Scroll forward.
PF8	Scroll backward.
PF12	Exit the Administration menu.

Option L – Node Name List

Option L (Node Name List) displays the node names of peer Data Object Brokers and their status.

```
HRNADML1  OSTARSRV              HURON NODE NAME LIST              2012AUG09 16:49:46

NO    NODE NAME          ALIAS              REQUESTED   ACTIVE   USER PREFIX
 0    HTSTSRV            HTSTSRV
 1    DEVSUN001          DEVSUN                   2         0          IRS
 2    DEVLX001           DEVLX                    2         0          IRS




                          ENTER-REFRESH  PF2-PROCESS NODE
```

Node Name List Fields

NO	The number of the connected peer
NODE NAME	The name of the connected peer
ALIAS	The alias of the connected peer
REQUESTED	Number of threads available
ACTIVE	Number of threads in use
USER PREFIX	The identifier prefix of the logged on user ID

Key Commands

PF1	Access Help information about the current screen.
PF2	Select a node. Place your cursor beside the desired NODE NAME and press PF2. All subsequent commands in Administration menu apply to the selected node until you exit the utility or use the NODE NAME LIST option to choose another node. The local node appears by default whenever you select the NODE NAME LIST option, regardless of previous node processing.
PF3	Return to the main menu.
PF6	Print the current screen.
PF11	Establish a connection to a peer node. Place the cursor beside the desired NODE NAME and press PF11.
PF12	Exit the Administration menu.

See Also *TIBCO Object Service Broker Application Administion* for more information about distributed data implementation and peer Data Object Brokers.

Chapter 5, Configuring a Communication Environment for Access of Distributed Data, on page 105 for more information about peer Data Object Broker configuration.

Option U – Huron Page Image

Option U (Huron Page Image) displays page images from an active Pagestore. The display consists of a page header, followed by a display in dump format.

```
HRNADMU1  OSTARSRV           HURON PAGE IMAGE DISPLAY        2012AUG09 15:41:38

PAGE 0000-00000372 PREV FFFFFFFF NEXT FFFFFFFF ROWS 0001 SIZE 01E1 TYPE DATA

     0000    00000000  0372FFFF  FFFFFFFF  FFFFC400   | .... .... .... ..D. |
     0010    93042814  0524011E  CF0000F3  000101E1   | l... .... ...3 .... |
     0020    00130400  0000010B  4BA285A3  A4974088   | .... .... .set up h |
     0030    85939700  28040000  0002204B  88F140E5   | elp. .... .... h1 V |
     0040    C9C5E640  D7D9D6C2  D3C5D440  D9C5D7D6   | IEW  PROB LEM  REPO |
     0050    D9E340C6  D6D3D3D6  E6E4D700  48040000   | RT F OLLO WUP. .... |
     0060    000340C4  85A38189  93858440  89958696   | .. D etai led  info |
     0070    999481A3  89969540  969540A3  8889A240   | rmat ion  on t his  |
     0080    97999682  93859440  89A24084  89A29793   | prob lem  is d ispl |
     0090    81A88584  40819396  958740A6  89A38840   | ayed  alo ng w ith  |
     00A0    A3888500  2D040000  00042589  95869699   | the. .... ...i nfor |
     00B0    9481A389  96954097  9996A589  84858440   | mati on p rovi ded  |
     00C0    899540A3  88854086  96939396  A6A4974B   | in t he f ollo wup. |
     00D0    000D0400  00000505  4BA29740  F1005404   | .... .... .sp  1... |
     00E0    00000006  4CC98640  A3888540  E285A585   | .... <If  the  Seve |
     00F0    9989A3A8  6B40D481  899540D9  85868599   | rity , Ma in R efer |

  PAGE/SEARCH TOKEN: _____________________________      (M FOR MAX SCROLL)
  PF2-READ  PF5-FIND  PF7-BACK  PF8-FORWARD  PF10-PREV PAGE  PF11-NEXT PAGE
```

Displaying a New Page Image

To display a new page image, enter the segment (*s*) and page number (*p*) in the **PAGE/SEARCH TOKEN** field. Use hex notation and a hyphen (-) to separate the segment and page:

```
ssss-pppppppp
```

and press PF2. If the segment number is omitted, the same segment number as the last display is used. Leading zeroes can be omitted from the segment number and the page number.

Specifying Search Tokens

Search tokens can be specified in one of the following ways:

- `x'token'`
- `c'token'`

- `'token'`
- `token`

The search function supports *first*, *last*, *previous*, and *next* type attributes (only the first letter is required).

Key Commands

PF1	Access Help information about the current screen.
PF2	Read a new page.
PF3	Return to the main menu.
PF5	Find a page according to a search token.
PF6	Print the current screen.
PF7	Scroll backward.
PF8	Scroll forward.
PF10	Previous page.
PF11	Next page.
PF12	Exit the Administration menu.

Option X – In-doubt Transactions

Option X (In-doubt Transactions) displays a list of transactions involving service providers that could not be fully completed for some reason. If the system is recycled before an in-doubt transaction is resolved, TIBCO Object Service Broker automatically reacquires the logical locks for the transaction during the restart process.

In-doubt Transactions List Screen

The In-doubt Transactions List screen displays a list of in-doubt transactions.

```
HRNADM11  OSTARSRV        IN-DOUBT TRANSACTIONS LIST        2012AUG09 16:58:34

                                       LOCAL                ORIGINATING
TRK     DATE    TIME    STAGE    STATUS  LOCK ID  USER ID  LOCK ID  NODE NAME
  1 2012AUG09 16:55 CONTNGT IN-DOUBT 00000918 USR40

                PLACE CURSOR ON DESIRED TRACK AND HIT PF2 TO ZOOM
```

In-doubt Transaction Fields

TRK	Track Number – identifies the contingency log entry.
DATE and TIME	The date and time when the commit was first saved on the contingency log.

STAGE	Stage or phase within the commit cycle that is currently being processed: PHASE 1 – Prepare-to-Commit stage of a two-phase commit. PHASE 2 – Commit stage of a two-phase commit. CONTINGT (Contingent) – TDS updates are held pending confirmation of the service provider's update status.
STATUS	Identifies the current status of the contingency log: PROGRESS – The commit cycle is continuing. M-COMMIT – The commit continues because of a manual intervention. M-ABORT – The commit is aborting because of a manual intervention. P-ABORT – The commit coordinator determined the commit cycle should abort; however, either all resources have not been informed, or have not confirmed their aborts. HELD – The local TDS concerns of a commit are done; however, there are service provider commit or cleanup confirmations outstanding. CHECKING – The commit coordinator is currently attempting to resolve an in-doubt transaction. IN-DOUBT – A service provider was lost during the commit cycle; it is undetermined if the commit should continue.
LOCAL LOCK ID	A reference number generated by the TIBCO Object Service Broker Data Object Broker to relate concerns within an identifiable transaction.
USERID	An eight-character identifier used by TIBCO Object Service Broker security to determine access privileges.
ORIGINATING LOCK ID and NODE NAME	If the contingency log was created as a result of a commit coordinated by a different Data Object Broker, the lock ID and node name displayed identify the transaction on the named TIBCO Object Service Broker node.

Key Commands

PF2	Displays a detailed list of all resources involved in the in-doubt transactions.

In-doubt Transaction Display Screen

The In-doubt Transaction Display screen displays details of all resources involved in in-doubt transactions. To display the screen, place the cursor on the desired track on the IN-DOUBT TRANSACTIONS LIST screen and press PF2. A screen similar to the following appears:

```
HRNADM12  OSTARSRV       IN-DOUBT TRANSACTIONS DISPLAY      2012AUG09 16:58:34

                                        LOCAL                ORIGINATING
TRK    DATE    TIME   STAGE   STATUS  LOCK ID  USER ID  LOCK ID  NODE NAME
  1 2012AUG09 16:55 CONTNGT IN-DOUBT 00000918 USR40

                          TRANSACTION RESOURCE LIST

                  CONFIGURED   PROCESSING                    ACCESS    REMOTE
  TYPE  SERVER ID F.S. LEVEL   F.S. LEVEL    NODE             ID      TRACK

   TDS   LOCAL
   HRN   HDBM4001  FAIL SAFE 2  FAIL SAFE 1   B              A2B00002




          PF2-TDS DETAIL  PF4-IN-DOUBT LIST  PF10-COMMIT  PF11-ABORT
```

Key Commands

PF1	Access Help information about the current screen.
PF2	Display local TDS tables that are held as locked by the transaction and display summary update counts against the table.
PF3	Return to the main menu.

PF4	Redisplay the In-doubt Transaction List screen.
PF6	Print the current screen.
PF10	Manually commit the transaction, if authorized.
PF11	Abort the transaction, if authorized.
PF12	Exit the Administration menu.

Option O – Local Diagnostics: Operator Command

Option O (Operator Command) displays the local Data Object Broker log file and provides a command line to enter operator commands. Initially the most recent log entries appear. The result of entering operator commands appears in the log file. If the Data Object Broker you are connected to is remote, you can issue operator commands; the log portion of the screen is blank.

```
HRNADMO1  OSTARSRV        LOG DISPLAY/OPERATOR COMMAND        2012AUG09 16:37:19
                  LOG FILE: E:\OSTAR32\log\hrncr.001
--------------------------------------------------------------------------------
2012/08/09 15:03:27  S6BUS002I hrnckpt startup completed
2012/08/09 15:03:27  S6BUS014I Starting hrnredo
2012/08/09 15:03:30  S6BUS002I hrnredo startup completed
2012/08/09 15:03:30  S6BUS014I Starting hrnappl(0)
2012/08/09 15:03:31  S6BUS002I hrnappl(0) startup completed
2012/08/09 15:03:31  S6BUS014I Starting hrnappl(1)
2012/08/09 15:03:31  S6BUS002I hrnappl(1) startup completed
2012/08/09 15:03:31  S6BUS014I Starting hrnappl(2)
2012/08/09 15:03:31  S6BUS002I hrnappl(2) startup completed
2012/08/09 15:03:31  S6BUS014I Starting hrncomt
2012/08/09 15:03:31  S6BUS002I hrncomt startup completed
2012/08/09 15:03:31  S6BUS014I Starting hrncomwq
2012/08/09 15:03:32  S6BUS002I hrncomwq startup completed
2012/08/09 15:10:36  S6BUA021I User 'SYSADMIN' logged on from 'OSTARSRV'
2012/08/09 15:11:23  S6BUA024I User 'SYSADMIN' logged off from 'OSTARSRV'
2012/08/09 15:11:36  S6BUA032I Operator 'abc00' logged on from 'OSTARSRV'
--------------------------------------------------------------------------------
  COMMAND -> ____________________________________________________________________
                    PF4-TOP OF LOG  PF5-BOTTOM OF LOG
          ENTER-REFRESH  PF7-BACK  PF8-FORWARD  PF10-LEFT  PF11-RIGHT
```

Key Commands

Enter	Refresh the display.
PF1	Access Help information about the current screen.
PF3	Return to the main menu.
PF4	Scroll to the top of the log display.
PF5	Scroll to the bottom of the log display.
PF6	Print the current screen.

PF7	Scroll back through the log display.
PF8	Scroll forward through the log display.
PF10	Scroll left in the log display.
PF11	Scroll right in the log display.
PF12	Exit the Administration menu.

Chapter 7 Using the Interface to TIBCO Hawk

This chapter describes how to use the interface to TIBCO Hawk.

Topics

Overview

The TIBCO Object Service Broker for Open Systems interface to TIBCO Hawk provides the same functionality that is available with the TIBCO Object Service Broker for z/OS interface to TIBCO Mainframe Service Tracker.

TIBCO Mainframe Service Tracker functions similar to a TIBCO Hawk microagent, exposing methods that are subscribed to by Hawk agents, allowing the agents to monitor a z/OS system. TIBCO Object Service Broker for z/OS has embedded Tracker Event Client capabilities, allowing it to provide event information to TIBCO Mainframe Tracker that in turn will be relayed to TIBCO Hawk agents through method subscriptions.

For Open Systems, a Hawk microagent that supports the same methods utilized by TIBCO Object Service Broker for z/OS has been embedded within the Data Object Broker, TIBCO Object Service Broker monitor process (osMon), and TIBCO Object Service Broker batch client (osBatch).

Features

The interface to TIBCO Hawk provides the following functionality:

- Allows monitoring for the initialization and termination of TIBCO Object Service Broker components.
- Enables rule-based applications to send event information to TIBCO Hawk.

See Also *TIBCO Mainframe Service Tracker Installation and Administration* and the TIBCO Hawk® documentation.

Enabling Hawk Microagents

To enable the Hawk microagent within a Data Object Broker, one or more of the HAWKDAEMON, HAWKNETWORK, and HAWKSERVICE Data Object Broker parameters must be set by editing the crparm configuration file. For information on editing the crparm configuration file, see Setting Data Object Broker Parameters, page 46.

To enable the Hawk microagent within a TIBCO Object Service Broker monitor process or TIBCO Object Service Broker batch client, one or more of the HAWKDAEMON, HAWKNETWORK, and HAWKSERVICE Execution Environment parameters must be set.

See Also *TIBCO Object Service Broker Parameters* for more information about the HAWKDAEMON, HAWKNETWORK, and HAWKSERVICE Data Object Broker and Execution Environment parameters.

Hawk Microagent Names

This section describes Hawk microagent application names and display names.

Application Names

A Hawk microagent application name is an identifier that can be used by TIBCO Hawk when applying a Hawk rulebase. It is used internally by TIBCO Hawk to identify a Hawk microagent running within a TIBCO Hawk managed application.

Application name collision can occur if more than one concurrently active Hawk-managed application has embedded Hawk microagents that are managed by the same Hawk agent and use the same application name. In this situation, when applying a rulebase, a Hawk agent will not be able to distinguish between the microagent instances.

The default application names are as follows:

- com.tibco.dob – the Hawk microagent embedded within the Data Object Broker.
- com.tibco.osMon – the Hawk microagent embedded within the TIBCO Object Service Broker monitor process.
- com.tibco.osBatch – the Hawk microagent embedded within the TIBCO Object Service Broker batch client.

To prevent application name collisions, the HAWKANAME Data Object Broker and Execution environment parameters can be used to uniquely identify a Hawk microagent instance.

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

Display Names

A Hawk microagent display name is a user-friendly name for a Hawk microagent that is seen when using a TIBCO Hawk console application such as the TIBCO Hawk Display.

The display name for the Hawk microagent embedded in a Data Object Broker, TIBCO Object Service Broker monitor process, or TIBCO Object Service Broker batch client can be set using the HAWKDNAME Data Object Broker and Execution Environment parameters.

The default display name for a Hawk microagent embedded in the Data Object Broker has the following form:

```
TIBCO OSB Data Object Broker, Name name, Host host
```

The default display name for a Hawk microagent embedded in the TIBCO Object Service Broker monitor process has one of the following forms:

```
TIBCO OSB Monitor, Name name, Host host, Port port
TIBCO OSB Monitor, Name name, Host host, IP address, Port port
```

The latter form is used when the TIBCO Object Service Broker monitor process is configured to only listen on a specific network interface using either the HOST or IP parameters.

The default display name for a Hawk microagent within a TIBCO Object Service Broker batch client has the following form:

```
TIBCO OSB Batch Execution Environment, Name name, Host host
```

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

Hawk Microagent Startup

When a Hawk microagent has been enabled, it is started as part of the initialization of the Data Object Broker, TIBCO Object Service Broker monitor process, or TIBCO Object Service Broker batch client. If the Hawk microagent is not able to connect to TIBCO Hawk during startup, it will make attempts to connect at regular intervals as specified by the HAWKPOLL parameter.

The HAWKPOLL parameter specifies the number of milliseconds that the Hawk microagent will sleep between connection attempts. If an internal event requires the Hawk microagent to send a message to TIBCO Hawk while the microagent is sleeping, a connection attempt will be performed at that time.

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

Hawk Microagent Methods

The Notification method of TIBCO Mainframe Service Tracker is the only method supported, since it is the only method utilized by the TIBCO Object Service Broker for z/OS interface to TIBCO Mainframe Service Tracker.

Note that all Hawk microagents expose a `_onUnsolicitedMsg` method, but this method is not actively supported, so subscriptions to this method will never receive any data.

Notification Method Parameters

The Notification method as exposed by TIBCO Mainframe Service Tracker and the Hawk microagents within TIBCO Object Service Broker for Open Systems has the method parameters described below:

Name	Type	Description
Message Number	string	Message identifier/number
Severity	string	Severity: I, W, E, C, F
Support Action	string	Support action
Issuer Source	string	Program, job name of issuer
Issuer Sub-source	string	Function of procedure
Issuer Correlation	string	Reference to the message origination
Host Name	string	Name of host operating system
Host Platform	string	Platform: Batch, CICS, IMS, Substation, OSB, USS
Message Text	string	Descriptive text

Component Initialization and Termination

Subscriptions to the Notification method can be used to be notified of the initialization and termination of TIBCO Object Service Broker for Open Systems Data Object Brokers and Execution Environments.

Data Object Brokers

To be notified of the initialization or termination of a Data Object Broker, subscribe to the Notification method available through the Hawk microagent embedded within that Data Object Broker.

Execution Environments (batch or online)

To receive initialization or termination notifications for a batch Execution Environment, subscribe to the Notification method available through the Hawk microagent embedded within the TIBCO Object Service Broker batch client.

Online Execution Environments are started by a TIBCO Object Service Broker monitor process. To receive initialization or termination notifications for online Execution Environments, subscribe to the Notification method available through the Hawk microagent embedded within the TIBCO Object Service Broker monitor process. Notification of an Execution Environment's initialization occurs when its first session is started; notification its termination occurs when it has been shutdown by the TIBCO Object Service Broker monitor process, which may be a result of the monitor process termination, or the Execution Environment being idle. Refer to the `IDLETIMEOUT` Execution Environment parameter in *TIBCO Object Service Broker Parameters* for details.

Notification Parameter Values

The following table details the values of the parameters of the Notification method on completion of initialization or start of termination, for a Data Object Broker, Online Execution Environment, and Batch Execution Environment.

Parameter	DOB	Online EE	Batch EE
Message Number	S6BHK100[1] / S6BHK200[2]		
Severity	I		
Support Action	empty		
Issuer Source	DOB node name	Execution Environment name	
Issuer Sub-source	OSB DOB	OSB Monitor	OSB Batch EE
Issuer Correlation	empty		
Host Name	host name		
Host Platform	OSB		
Message Text	Initialization Complete[1] / Termination in Progress[2]		

[1]Component initialization
[2]Component termination

S6BNOTIFY Shareable Tool

The S6BNOTIFY shareable tool allows rules-based applications to send Notification messages to TIBCO Hawk.

To be notified as a result of a call to the S6BNOTIFY shareable tool in an online Execution Environment, a subscription must be made to the Notification method available through the Hawk microagent embedded within the TIBCO Object Service Broker monitor process that started the Execution Environment.

To be notified as a result of a call to the S6BNOTIFY shareable tool in a batch Execution Environment, a subscription must be made to the Notification method available through the Hawk microagent embedded within the TIBCO Object Service Broker batch client.

See Also *TIBCO Object Service Broker Shareable Tools* for details on S6BNOTIFY.

Appendix A Sample Configurations

This appendix describes a sample installation and some sample configurations to help you set up your own TIBCO Object Service Broker system.

Topics

A Sample Installation

The large number of parameter files in an TIBCO Object Service Broker installation are designed to increase flexibility and give system administrators greater control. The following sample installation should help you understand the connections among these files.

This installation has 100 users of which 15 are developers, and 85 are production users. Each of these groups needs a separate service from TIBCO Object Service Broker. This system comprises three production Data Object Brokers connected as distributed data peers, with the developers using a fourth Data Object Broker, in Perth. Production is spread over three locations: Perth, Sydney, and Melbourne. The system administrator wants to be able to shutdown and restart any of these cities' servers as a group, and to be able to identify the servers by the names appearing in the Data Object Broker log.

Data Object Broker

The first step is to identify the Data Object Brokers. You do this using parameters in the crparm file and the huron.dir file.

This is part of the crparm file for the Data Object Broker serving the users in Perth:

```
NODENAME=PDOB
```

The huron.dir file for this Data Object Broker appears as follows:

```
node name=PDOB,
     host=WINHOST,
     port=12000
```

TIBCO Object Service Broker Monitor Process

The next step is to create a separate TIBCO Object Service Broker monitor process (osMon) for each Data Object Broker. Multiple osMon's can run on the same machine as long as each has a unique port and tn3270port number. An identical mon.prm file is defined for each of Perth, Sydney and Melbourne. This is the resulting mon.prm file for our installations:

```
NAME=DEV
DOB=MYDOB
PORT=9068
TN3270PORT=9099
MONLOG=d:\ostar\log\devmon.log

NAME=PERTH
DOB=PDOB
SERVERS="10 PERSERV"
PORT=9069
TN3270PORT=9010
MONLOG=d:\ostar\log\permon.log

NAME=SYDNEY
DOB=SDOB
SERVERS="10  SYDSERV"
PORT=9070
TN3270PORT=9011
MONLOG=d:\ostar\log\sydmon.log

NAME=MELBOURNE
DOB=MELDOB
SERVERS="10  MELBSERV"
PORT=9071
TN3270PORT=9012
MONLOG=d:\ostar\log\melmon.log
```

Server Group Identification

Each group of SERVERS referenced in mon.prm requires a NAME entry in the `session.prm` file:

```
NAME= PERSERV
SERVERTYPE=PRS
SEARCH=I
EENAME=PERTH
SERVERLOGPATH=d:/temp

NAME= SYDSERV
SERVERTYPE=PRS
SEARCH=I
EENAME=SYDNEY
SERVERLOGPATH=d:/temp

NAME= MELSERV
SERVERTYPE=PRS
SEARCH=I
EENAME=MELBOURNE
SERVERLOGPATH=d:/temp
```

The same session.prm file can be used for each location. For ease of identification, we specified a unique EENAME for each location. You use the EENAME to reference a NAME entry in the ee.prm file, discussed later.

Setting Up Parameters for Users

Generally, the users of this system use the TIBCO Object Service Broker UI client, for which a parameter file is not required. The user can specify session parameters when starting the TIBCO Object Service Broker UI.

To log in using a 3270 emulator, a user supplies the host name or IP address of the machine running osMon and the osMon tn3270port number. For example, to log in to the Data Object Broker in Sydney, a user specifies the host name or IP address of the Sydney machine and a tn3270port of 9011.

Identifying the Execution Environment

Finally, an *ee.prm* entry must be created for each of the EENAMEs used in the session.prm.

```
NAME=PERTH
 MAXSESSION=25
 SECAUDITLOG=DISABLED
 EELOG=d:/TIBCO Object Service Broker/log/PERTHee.log

NAME=SYDNEY
 MAXSESSION=25
 SECAUDITLOG=DISABLED
 EELOG=d:/TIBCO Object Service Broker/log/PERTHee.log

NAME=MELBOURNE
 MAXSESSION=25
 SECAUDITLOG=DISABLED
 EELOG=d:/TIBCO Object Service Broker/log/MELBee.log
```

Each Execution Environment has MAXSESSION set to 25. This does not mean that a total of 25 sessions can be supported by each Execution Environment. MAXSESSION controls the total number of sessions associated with a particular instance of that Execution Environment. Every Execution Environment can have multiple instances defined, up to the number specified by the MAXEE Execution Environment parameter. By not specifying a value for MAXEE in mon.prm, we choose to use the default value of 100. So in our example, the maximum number of sessions per Execution Environment is 100 x 25 (MAXEE x MAXSESSION) = 2500.

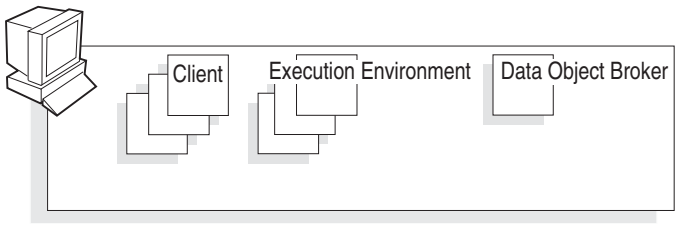
Using these configurations, the administrator can start and stop a distributed database server with a command such as:

```
osMon name=perth
...
osMon name=perth stop
```

Sample Configurations

Standalone Configuration

In this configuration, all TIBCO Object Service Broker components are installed on one machine. Use this configuration for TIBCO Object Service Broker administrators who require an initial test system or application developers who require a standalone system.

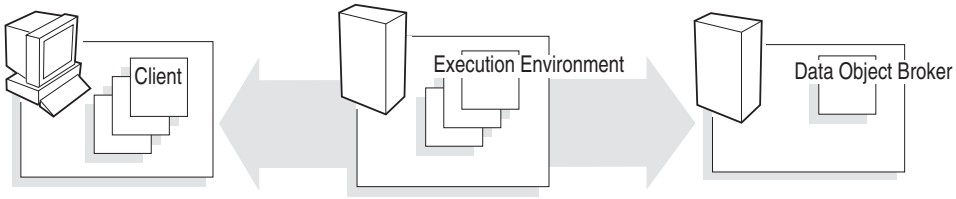


You could use parameter files such as these with this configuration:

tty.prm	mon.prm	huron.dir	crparm
NAME=DEFAULT MONPORT=9068 MONHOST=WINhost	NAME=DEFAULT DOB=LocalDOB PORT=9068	node name=LocalDOB, host=WINhost, port=12000	NODENAME= LocalDOB

Sample User Application System (Thin Client)

In this configuration, clients are installed on workstations and Execution Environments are installed on server machines.

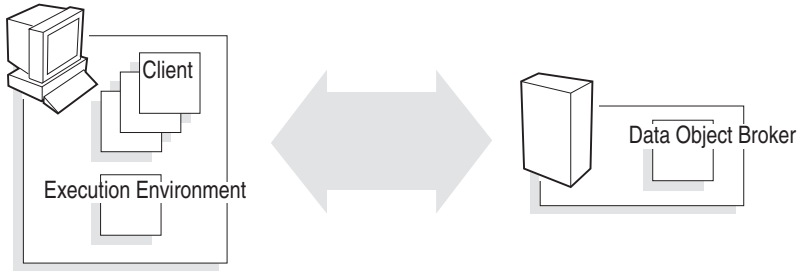


You could use parameter files such as these with this configuration:

On the client machine:	On the Execution Environment (EEHost) machine:		On the Data Object Broker machine:	
tty.prm	mon.prm	huron.dir	crparm	huron.dir
NAME=DEFAULT MONPORT=9068 MONHOST=EEhost	NAME=DEFAULT DOB=WINDOB PORT=9068	node name=WINDOB, host=WINhost, port=12000	NODENAME=WINDOB	node name=WINDOB, host=WINhost, port=12000

Sample User Application System (Fat Client)

In this configuration, the Execution Environments reside on the client machine. Having the sessions on the same machine can be advantageous for the application development cycle.



You could use parameter files such as these with this configuration:

On the client/Execution Environment machine:			On the Data Object Broker machine:	
tty.prm	mon.prm	huron.dir	crparm	huron.dir
NAME=DEFAULT MONPORT=9068 MONHOST=EEhost	NAME=DEFAULT DOB=WINDOB PORT=9068	node name=WINDOB, host=WINhost, port=12000	NODENAME=WINDOB	node name=WINDOB, host=WINhost, port=12000

Appendix B Possible Problems

This appendix describes TCP/IP subnet connection problem and DLL initialization failure on Windows.

Topics

TCP/IP Subnet Connection Problem

Router Cannot Send Back ICMP Messages

On Windows, the TCP/IP stack does not fragment data into a minimum packet size when transmitting data between subnets. It relies on the routers to do this, but the routers must be configured correctly to send an ICMP message back to Windows acknowledging that they cannot pass the data over since it is too large. In response, the stack transmits the data in smaller packets. The stack behaves in this manner by default to increase network response.

Symptoms

If you have dumb routers that are unable to send back ICMP messages, and if the stack is running in default mode, you are likely to experience problems using **`ftp`** to send data over the network. This can result in connection problems between TIBCO Object Service Broker component machines in different subnets.

Workaround

Modify the Windows registry on affected machines on both subnets. Complete the following steps:

1. Invoke the registry editor (regedt32.exe).
2. Select the HKEY_LOCAL_MACHINE on Local Machine Window.
3. Select SYSTEM (double click).
4. Select CurrentControlSet.
5. Select Services.
6. Select TCP/IP.
7. Select Parameters.
8. Select the Edit menu and choose Add value.

   The Add Value dialog appears.
9. Type EnablePMTUDiscovery (case sensitive) in the Value Name field.
10. Select REG_DWORD in Data Type and click the OK button.
11. The DWORD editor dialog appears. Type the data value 0.
12. Save the changes.
13. Shutdown and restart your machine for the service to be in effect.

DLL Initialization Failure (Windows)

osee DLL Does Not Initialize

When starting a session, you can get a message that the DLL called IMAGEHLP.DLL failed to initialize. At that point, the process terminates.

Possible Cause

You could need more memory in your Windows paging file.

Solution

Increase the virtual memory for your Windows. To change the size, do the following:

1. On the Start menu, navigate to and start the Control Panel.
2. Double-click the System icon.
3. Click the Performance tab.
4. In the Virtual Memory section, click the Change button.
5. Increase the number in the Maximum Size field.

You have to increase it only until the DLL can initialize, possibly to 300 (MB).

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W