TIBCO Service Gateways™ for ODBC and Oracle

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. You can use TIBCO Object Service Broker to access external DBMS data and define TIBCO Object Service Broker tables based on this data. This manual describes the TIBCO Service Gateway for ODBC and TIBCO Service Gateway for Oracle interfaces to external DBMS 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 Introduction

This chapter provides an overview of the TIBCO Service Gateway for ODBC and the TIBCO Service Gateway for Oracle.

Topics

Overview of the Gateways

What are TIBCO Service Gateway for ODBC and TIBCO Service Gateway for Oracle?

The Gateways are server processes that you can use to access external DBMS data from a TIBCO Object Service Broker session.

You use the TIBCO Service Gateway for ODBC to access ODBC data from within TIBCO Object Service Broker and you use TIBCO Service Gateway for Oracle data to access Oracle data from within TIBCO Object Service Broker.

Both Gateways run in a TIBCO Object Service Broker Execution Environment under Windows. TIBCO Service Gateway for Oracle can also run under Solaris and Linux. The Gateways ensure that data is presented to TIBCO Object Service Broker rules in a manner consistent with TIBCO Object Service Broker behavior. The Gateways convert TIBCO Object Service Broker requests into ODBC API function calls and Oracle OCI function calls, and use these APIs and OCIs to communicate with the external DBMSs.

Because both Gateways exhibit similar behavior and yield similar functionality, they are described in this manual as one component, with their differences also indicated.

The Gateways are installed with TIBCO Object Service Broker.

Components of the Gateways

The Gateways consist of the following components:

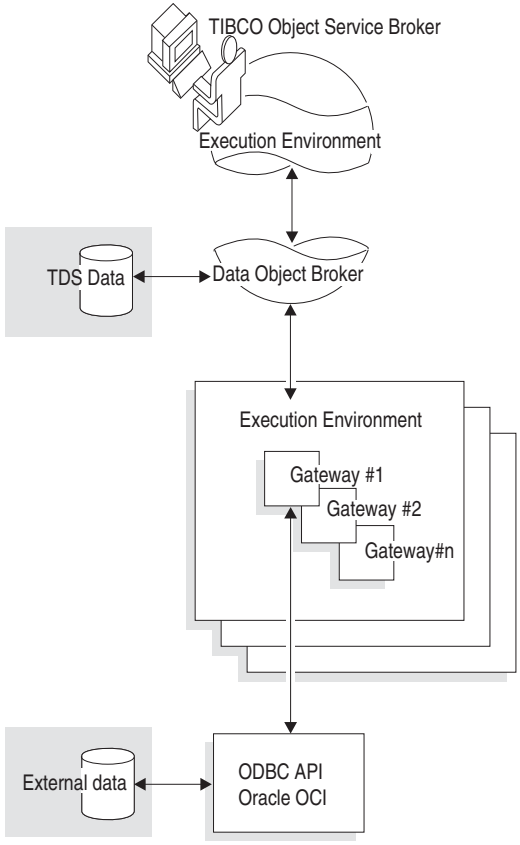
Component	Function
Table Definer	• Define TIBCO Object Service Broker tables of type SLK • Define stored procedures only for TIBCO Service Gateway for ODBC
The Gateways	Access external DBMS definitions and data using SLK tables.
Metadata Extractor	Extract table definitions from external DBMSs and store them in TIBCO Object Service Broker TDS tables.

Accessing TDS and External DBMS Data

Using the Gateways' components, you access external DBMS data by:

1. Configuring the Execution Environment
2. [z/OS only] Setting up resources for your Gateways through the TIBCO Object Service BrokerAdministration Menu
3. Starting one or more Gateways to convert requests into ODBC API or Oracle OCI API function calls
4. Defining an SLK table based on an external table definition and optionally mapping stored procedures with TIBCO Service Gateway for ODBC
5. Using the Table Browser, the Table Editor, and the rules language to manipulate the data

The following diagram illustrates the access flow.



Overview of the ODBC API and Oracle OCI

What is ODBC?

ODBC is a generic API for accessing a wide range of DBMSs, including Oracle, DB2, Sybase, Ingres, Informix, Microsoft SQL Server, Progress, and others. You use the unified interface to access data stored in SQL-based databases and in ODBC-enabled applications, like Excel, which do not require data source processing. Because of the generic nature of the ODBC API, you can use ODBC with TIBCO Object Service Broker as a Gateway to external DBMS data on the Windows platform.

What is Oracle OCI?

OCI is Oracle's Call Interface. You can use the Oracle OCI to interact with one or more Oracle servers in performing the full range of database operations such as queries, locking, modification, and transaction handling.

The Gateways and TIBCO Object Service Broker SLK Tables

The Gateways are invoked whenever a request to access a table of type SLK is detected by the Data Object Broker. An SLK table maps an external table so that it can be accessed via a Gateway.

"SLK" is used to denote the TIBCO Object Service Broker table type used by both Gateways. SLK is also the SERVERTYPE attribute for TIBCO Service Gateway for ODBC, whereas the SERVERTYPE for TIBCO Service Gateway for Oracle is ORS.

Refer to Defining an SLK Table on page 48 for more information.

SLK Table Definition for Access to ODBC Data

An SLK table definition contains the following information, common to all external DBMSs accessible through ODBC:

- Server type (SLK)
- Server ID
- External DBMS table name
- Qualifier (also known as catalog)
- Owner (also known as schema)

- List of fields (name, data type, precision, and scale)
- ODBC data source name

SLK Table Definition for Access to Oracle Data

An SLK table definition contains the following information:

- Server type (ORS)
- Server ID
- Oracle table name
- Qualifier (known in Oracle as DBlink)
- Owner
- List of fields (names, data type, precision, scale)
- Oracle SID

TIBCO Object Service Broker Table Definitions

SLK table definitions use the following TIBCO Object Service Broker information when mapping to external DBMS tables:

- Table name
- Location
- Event rules
- Fields (name, semantic type, syntax, length, decimals, and default value) required to map the external fields

ODBC and Data Sources

The concept of the data source is fundamental to the ODBC approach to generalized data access. Data sources are stored in a repository maintained by the Windows operating system. At login time, TIBCO Service Gateway for ODBC makes use of the data source specified in the SLK table definition to connect to the appropriate DBMS instance. Therefore, a data source name is used for retrieving login information without having to hardcode it within a table definition.

ODBC Data Sources

The following information is stored in an ODBC data source:

- Data source name

- Reference to the ODBC driver to be invoked
- Optional DBMS or driver-specific parameters

After defining a table in TIBCO Object Service Broker, you can modify the contents of the corresponding data source so that a subsequent request is directed to a different database without altering the table definition. However, this request can be successful only if the external DBMS contains a table mapped by the TIBCO Object Service Broker definition.

Oracle SID

The Oracle SID identifies the Oracle server to log on to so you can access Oracle data. It plays the same role for TIBCO Service Gateway for Oracle as ODBC Data Source does for TIBCO Service Gateway for ODBC.

Functional Overview of Data Access

How the Gateway Transactions Process External DBMS Data

The process used by TIBCO Object Service Broker to access data via its external Gateways, consists of the following basic steps:

1. The Data Object Broker receives a client request containing a table definition, which, in turn, contains a table type.
2. The Data Object Broker selects a Gateway (considered to be a shared resource) by matching the SERVERTYPE and SERVERID values in the request to a corresponding Gateway.
3. The Data Object Broker sends the request to the Gateway, which converts the request into the form expected by the external DBMS and then submits the transformed request.
4. Data returned by the external DBMS is converted into TIBCO Object Service Broker format, sent to the Data Object Broker, and then to the client session.

Configuring Execution Environments

Before you start the Gateway, you must configure the Execution Environment where the Gateway is to run.

For more information about configuration, refer to Chapter 3, Configuring and Operating, on page 25.

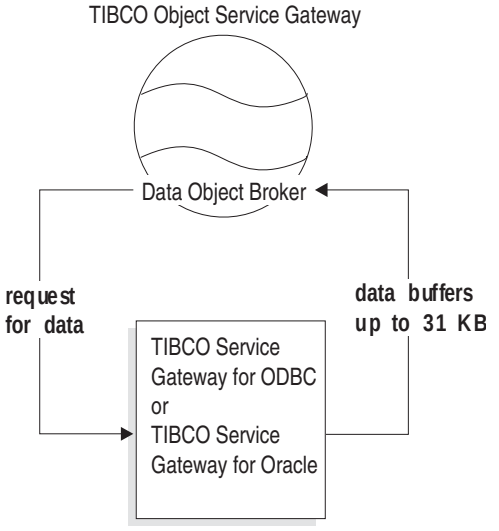
Security Settings for Data Access

The setting of the SECLEVEL gateway parameter dictates what security information the Gateway uses to connect to an external DBMS when access to external data is requested. For example, if SECLEVEL=0, the Gateway uses the TIBCO Object Service Broker user ID and password to connect to an external DBMS. If SECLEVEL=1, the Gateway uses the TIBCO Object Service Broker client session ID, current group name, or an externally defined ID to connect to an external DBMS.

Sending Data to the Data Object Broker

Starting the Gateway creates a connection to the Data Object Broker. The Gateway maintains a connection with the Data Object Broker for the life of the entire Gateway session. Data is sent to the Data Object Broker in variable length buffers up to a maximum of 31 KB. If a single request requires more than 31 KB of data, multiple 31 KB buffers are sent until the request is complete.

Data Sent to the Data Object Broker in 31 KB Buffers

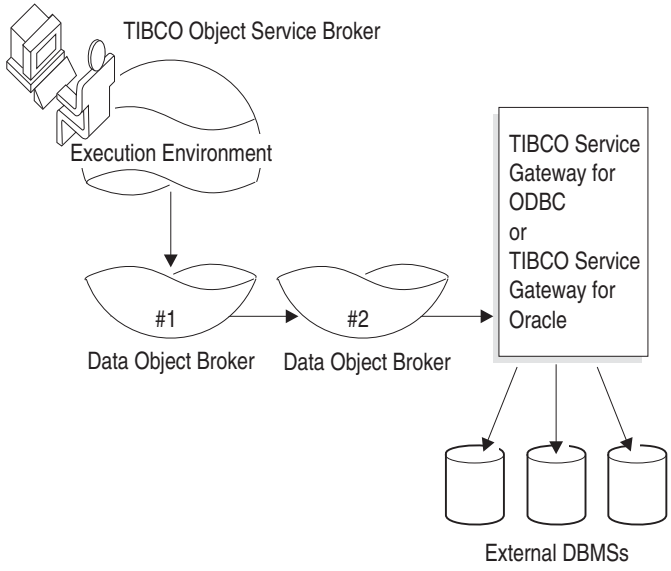


See Also The *External Environments* manual for your operating environment for information on configuring the Execution Environment

Using Distributed Data with the Gateways

Distributed access between TIBCO Object Service Broker and external DBMSs is permitted subject to the requirements of all distributed access. Refer to *TIBCO Object Service Broker Application Administration* and the *TIBCO Object Service Broker Installing and Operating* manuals for more information.

This illustration shows a sample Gateway distributed-data scenario:



Chapter 2 Installing

This chapter explains how to install Service Gateway for ODBC (Open Systems) and Service Gateway for Oracle (Open Systems).

Topics

Installer Overview

This section provides an overview of the installation process that applies to both Service Gateway for ODBC (Open Systems) and Service Gateway for Oracle (Open Systems). Hereafter, these products are referred to as simply Service Gateway for ODBC and Service Gateway for Oracle.

Local or Remote Installation

For the Service Gateway, the terms *local installation* and *remote installation* are used in relation to the presence of a Data Object Broker component of a TIBCO Object Service Broker installation. You can install the Service Gateway as follows:

- **Local (Enablement)** — The software is installed on the same host in which TIBCO Object Service Broker for Open Systems is installed. Both components share configuration files.
- **Remote on Separate Host (Standalone)** — The software is installed on a host in which TIBCO Object Service Broker for Open Systems is *not* present.
- **Remote on Same Host (Standalone)** — The software is installed on a host in which TIBCO Object Service Broker for Open Systems is present, but the components do *not* share configuration files. This installation type is suitable for implementations in which the Service Gateway communicates with a Data Object Broker other than that on the same host.

When you run the installer, it detects the presence of TIBCO Object Service Broker for Open Systems if it is installed on the target host. At that time, you are given the choice to install the software with the base product (shared configuration files), or separately from the base product (components do not share configuration files).

For Linux, the only supported installation type for Service Gateway for Oracle is a standalone installation, in which the Data Object Broker runs remotely on a separate host.

Installation Modes

You can run the installer in one of these three 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 line. This is useful if your machine does not have a window environment.
- **Silent mode** — Silent mode either installs using the default settings or uses a *response file* that contains properties you can set for your installation. Silent mode installs without prompting you for information. For details, see Generating and Using a Response File on page 21.

Preparing for Installation

This section describes the preparation for installing, and applies to both Service Gateway for ODBC and Service Gateway for Oracle.

Prerequisite Software

For supported software versions, see the readme file.

To run the installer, you must first install 32-bit Java Runtime Environment (JRE) or Java Development Kit (JDK) from the Oracle Technology Network.

Note that the `JAVA_HOME` environment variable must point to the directory that contains the 32-bit JRE installed from the Oracle Technology Network.

For Linux installations, where an OpenJDK version of Java may have been installed as part of the operating system, you must configure `JAVA_HOME` to point to the correct JRE installation.

Installer Disk Space Requirements

Windows

The package is extracted into a temp folder, typically `SystemDrive:\Temp` or `SystemDrive:\Documents and Settings\<user_name>\Local Settings\Temp`. The installer requires 60MB of free space in the temp directory.

Unix

When a regular (non-root) user installs a TIBCO 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 500 KB of free disk space.

Installer Account

Windows

You must have administrator privileges for the machine on which the Service Gateway 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.

If you intend to install the product on a network drive, you must ensure that the account used for installation has permission to access the network drive.

When installing on Windows terminal server, there are two modes: `Execute` and `Install`. By default all users are logged on in `Execute` mode, which allows them to run the applications. When you want to install the Service Gateway for use by everyone, the Administrator should change to `Install` mode.

The best way to install the Service Gateway is to use the Add/Remove Programs control panel applet, because this automatically sets the mode to `Install` during the installation and then back to `Execute` at the end. Alternatively, you can manually change your mode to `Install` by typing:

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

Change back to execute:

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

Check your current mode:

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

If you install in the `Execute` mode, the installation registry is maintained in your user home directory. If you install in the `Install` mode, the installation registry is maintained in the *%SystemRoot%* folder.

You must invoke the installer from a TIBCO Object Service Broker-enabled `CMD` prompt or from a `CMD` shortcut created by the base installer, in which the TIBCO Object Service Broker environment has already been set.

Unix

The Service Gateway can be installed by a regular (non-root) user and super-user (root). Different users can install the same product at different locations.

Installing Service Gateway for ODBC

Service Gateway for ODBC is only available on the Windows platform. Before starting installing, ensure that your system meets the hardware and software requirements, and that you have reviewed the pre-installation steps.

Installation File

The following is the installation file for the Service Gateway for ODBC:

```
TIB_srvcgw_odbc_version_win_x86.exe
```

where *version* is the current release number of the Service Gateway. For example, for software release 6.0.0, the file name is:

```
TIB_srvcgw_odbc_6.0.0_win_x86.exe
```

Installation Process

After starting the installer and accepting the license agreement, you are prompted to specify the *TIBCO_HOME* directory if one is not detected by the installer. Typically, this is the top level installation directory for all TIBCO products; however, you can specify a different directory if desired.

Install Service Gateway for ODBC in one of the following modes.

GUI Mode

This mode allows you input values in panels. To start the installation, type the following at the command prompt:

```
TIB_srvcgw_odbc_6.0.0_win_x86.exe
```

Console Mode

This mode allows you to install the software in a non-windows environment. The installer will prompt you for values. To start the installation, type the following at the command prompt:

```
TIB_srvcgw_odbc_6.0.0_win_x86.exe -console
```

Silent Mode

In this mode, the installer proceeds without prompting you for information. See Generating and Using a Response File on page 21 for details.

Post Installation Steps

After installing Service Gateway for ODBC, you must:

1. Configure the gateway by creating configuration files in the *TIBCO_HOME*\OSBgateways\6.0\database directory. For details, refer to Supplying the Gateway Startup Parameters, page 27.
2. Modify the batch file launchODBCgw.bat, in the *TIBCO_HOME*\OSBgateways\6.0\utils directory.

   Edit the file to supply osMon with a NAME matching the NAME parameter value you choose for Service Gateway for ODBC in mon.prm.

Installing Service Gateway for Oracle

Before installing the product, ensure that your system meets the hardware and software requirements, and that are listed in Preparing for Installation on page 14.

Installation Files

The following are the installation files for the Service Gateway for Oracle:

Windows

```
TIB_srvcgw-oracle_6.0.0_win_x86.exe
```

Solaris

```
TIB_srvcgw-oracle_6.0.0_sol.bin
```

Linux

```
TIB_srvcgw_oracle_6.0.0_linux26gl25_x86.bin
```

Installation Process

After starting the installer and accepting the license agreement, you are prompted to specify the *TIBCO_HOME* directory if one is not detected by the installer. Typically, this is the top level installation directory for all TIBCO products; however, you can specify a different directory if desired.

Install Service Gateway for Oracle in one of the following modes.

GUI Mode

This mode allows you input values in panels. To start the installation, type the following at the command prompt:

Windows

```
TIB_srvcgw_oracle_6.0.0_win_x86.exe
```

Solaris

```
./TIB_srvcgw_oracle_6.0.0_sol.bin
```

Linux

```
./TIB_srvcgw_oracle_6.0.0_linux26gl25_x86.bin
```

Console Mode

This mode allows you to install the software in a non-windows environment. The installer will prompt you for values. To start the installation, type the following at the command prompt:

Windows

```
TIB_srvcgw_oracle_6.0.0_win_x86.exe -console
```

Solaris

```
./TIB_srvcgw_oracle_6.0.0_sol.bin -console
```

Linux

```
./TIB_srvcgw_oracle_6.0.0_linux26gl25_x86.bin -console
```

Silent Mode

In this mode, the installer proceeds without prompting you for information. See Generating and Using a Response File on page 21 for details.

Post-Installation Steps

After installing Service Gateway for Oracle, you must perform the following tasks.

Windows

Follow these post-installation steps on Windows platforms:

1. Configure the gateway by creating configuration files in the *TIBCO_HOME*\OSBgateways\6.0\database directory. For details, refer to Supplying the Gateway Startup Parameters on page 27.
2. Modify the batch file launchOracle.bat, in directory *TIBCO_HOME*\OSBgateways\6.0\utils.

   Edit the file to to supply osMon with a NAME matching the NAME parameter value you choose for Service Gateway for Oracle in mon.prm.

UNIX

Follow these post-installation steps on Unix platforms:

1. Configure the gateway by creating configuration files in the directory *TIBCO_HOME*/OSBgateways/6.0/database. For details, refer to Supplying the Gateway Startup Parameters on page 27.

2. Modify the shell script `launchOraclegw`, in directory *TIBCO_HOME*`/OSBgateways/6.0/utils`.

   Edit the script to supply osMon with a NAME matching the NAME parameter value you choose for Service Gateway for Oracle in `mon.prm`.

Generating and Using 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).

This section describes how to generate and install with a response file.

Generating a Response File

To use silent 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` *filename*

where *installer* is the installer executable (or bin file on Solaris or Linux), and *filename* is the name of the response file to be generated.

For example, on Windows:

`TIB_srvcgw_oracle_6.0.0_win_x86.exe -options-record` *filename*

After invoking the installer, complete the installation as normal. Your installation selections are saved in the file specified in *filename*.

Installing in Silent Mode

Silent mode installs using a response file that contains properties you can set for your installation. Silent mode installs without prompting you for information.

You must have a previously generated response file to use this mode. See Generating a Response File above for details.

To run the installer in silent mode, run the installer using the following command:

installer `-silent -options` *filename*

where *installer* is the installer executable (or bin file on Solaris or Linux), and *filename* is the name of the response file that was previously generated.

For example, on Windows:

`TIB_srvcgw_oracle_6.0.0_win_x86.exe -silent -options` *filename*

Installation in GUI or Console Mode With Response File

A previously generated response file may optionally be used with the installer in GUI or console mode, in which case the response file determines the defaults that are presented.

To use a response file in GUI or console mode, invoke the installer from the command line using the following commands:

- GUI Mode:

  ```
  TIB_srvcgw_odbc_6.0.0_win_x86.exe -options filename
  ```

- Console Mode:

  ```
  TIB_srvcgw_odbc_6.0.0_win_x86.exe -console -options filename
  ```

where *filename* is the name of the response file that was previously generated.

Combination of Options

You can combine the different available options. For example, to install using Console mode and generate a response file, use:

```
TIB_srvcgw_odbc_6.0.0_win_x86.exe -console
                                  -options-record filename
```

Installation Choices

You have two installation choices, enablement and standalone, which apply to all installation modes.

Enablement Installation

To make the enablement choice available for installation, you must have installed a TIBCO Object Service Broker base system. Call the installer program from a TIBCO Object Service Broker-enabled `CMD` prompt or from a `CMD` shortcut created by the base installer, in which the `HURON` environment variable is already set. When the installer prompts you to choose between enablement and standalone, choose enablement.

The absence of a choice means that standalone installations already exist in the same `TIBCO_HOME` folder.

Standalone Installation

Standalone installation is the default in all other cases if no TIBCO Object Service Broker base system has been installed. This type of installation applies regardless of whether or not other standalone installations exist in the `TIBCO_HOME` folder.

Be extra-careful if you use silent mode because the target installation environment plays a primary role in allowing the installation types that are available.

Uninstalling the Software

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

Note that the uninstaller deletes all the directories created by the installer. The only exception to this rule is if a new file is added to a directory after installation. In that case, the directory is left intact and you can delete it at your discretion.

Windows

Use one of the following to uninstall Service Gateway for ODBC:

- Open the Programs and Features dialog box in the Control Panel.
- Navigate to the following directory:

  TIBCO_HOME\OSBgateways\6.0\uninstaller_archives\osbgateway_odbc\

  and invoke the `uninstall.exe` program.

UNIX

To uninstall Service Gateway for Oracle, navigate to the following directory and invoke the `uninstall.bin` program:

TIBCO_HOME/OSBgateways/6.0/uninstaller_archives/osbgateway_oracle/

Chapter 3 Configuring and Operating

This chapter provides the information to install, configure, start, shut down, and operate TIBCO Service Gateway for ODBC and TIBCO Service Gateway for Oracle.

Topics

Configuration Overview

Before You Configure the Gateways

Before using the Gateways, you must:

1. Install the ODBC client software or the Oracle client software you intend the Gateway to use.
2. Download and install the TIBCO Object Service Broker Release software available from our http://download.tibco.com web site.
3. Authorize access to the external data that you require.

Authorizing Access to External DBMS Data

The Gateways provide two methods of authorizing access to external DBMS data. The access method is determined by the SECLEVEL (Security Level) parameter specified at Gateway startup. Refer to Supplying the Gateway Startup Parameters on page 27 for more information.

SECLEVEL=0	External DBMS table access is verified using the SESSIONUSERID and SESSIONPASSWORD gateway startup parameters. The user ID and password, which are used to connect to the Gateway for all transactions, require the security described in the documentation provided with the external DBMS.
SECLEVEL=1	The Gateway table access is verified using the TIBCO Object Service Broker client session ID and password, current TIBCO Object Service Broker group name, or an externally defined user ID and password.

See Also *TIBCO Object Service Broker Managing Security* about TIBCO Object Service Broker security.

Supplying the Gateway Startup Parameters

The gateway parameters, which are described in the following section, are configurable via the `mon.prm` and `session.prm` parameter files used for TIBCO Object Service Broker sessions.

Configuring a Service Gateway to Run on Open Systems

To configure a Service Gateway, you must edit the following parameter files:

- `mon.prm`
- `session.prm`

The parameter files will be located in `%OS_ROOT%\database` on Windows and `${OS_ROOT}/database` on Solaris and Linux. Sample template files are provided in these directories (they have the extension `.template`).

If the parameter files already exist in you database directory, modify each file as required for installation. If the parameter files do not already exist, perform the following procedure:

1. Make a copy of each template file.
2. Modify each copy as required for your installation.
3. Rename the copies as appropriate to `mon.prm` or `session.prm`.

mon.prm File

A `mon.prm` file can include the following Execution Environment parameters used for accessing external servers:

SERVERS (SRV)

A double-quoted list of pairs of values, separated by commas. The first value in the pair is the number of Gateways to start and the second is the name of the group in the `session.prm` file describing the Gateway session.

For example, if the `mon.prm` file contains the following line: `SERVERS "1 ABC,3 DEF"`, the line notifies the TIBCO Object Service Broker monitor process that 1 Gateway described by the section named ABC and 3 Gateways described by the section named DEF are to be started at Execution Environment startup. The names ABC and DEF are references to the session.prm file where groups of session parameters with NAME ABC and NAME DEF as section headers are to be found.

SERVERRETRIES (SRT)

A number indicating how many times the attempt to start an external Gateway is to be repeated by the TIBCO Object Service Broker monitor process at Execution Environment startup. Set this parameter to 1 so that you are not retrying a Gateway startup without removing the cause of the failure.

session.prm File

A `session.prm` file describes Gateway sessions the same way it describes regular sessions: a named group of parameters can contain both generic session parameters and Gateway-specific session parameters. This Gateway-specific group should be referenced by the SERVERS Execution Environment parameter in the `mon.prm` file described above.

See Also *TIBCO Object Service Broker Parameters* for more information about the parameters and parameter files.

The Gateway Parameters

This section identifies the gateway parameters, their valid abbreviations, and their functions. To change the gateway parameters for an SLK table, refer to Supplying the Gateway Configuration Parameters on page 33.

EXTERNALUSERID (EUID)	Has meaning only if Security level=1. Specifies how the authorization ID is identified to the external DBMS. The default is USERID. Valid values: USERID – The TIBCO Object Service Broker Gateway session ID and password. GROUP – The current TIBCO Object Service Broker security group. The name of this group can be up to 16 characters long, but only 8 characters are supported. A SECURITYFAIL occurs for more than eight characters. **Note** A TIBCO Object Service Broker security group has no password associated with it, so only an external ID with an empty password can be used. OTHER – Requires an externally defined user ID (OTHERUSERID) and password (OTHERPASSWORD).
FSLEVEL (FSL)	Use to specify the level of Fail Safe processing. The default value is zero. Valid values: 1 – Activate Fail Safe Level 1. The Gateway informs the Data Object Broker that it can support Fail Safe level-1 processing. If the Gateway is to attach to a z/OS Data Object Broker, the Data Object Broker's connection attribute setting "commit level" must be set to 1. If not, the Gateway connection is rejected. Refer to Implementing Fail Safe Processing on page 38 for more information. You must specify the TRXDB, FSTABLENAME, RECOVERYID, and RECOVERYPASSWORD parameters. 0 – De-activate Fail Safe processing. The Gateway informs the Data Object Broker that it does not support Fail Safe level-1 processing. If the Gateway is to attach to a z/OS Data Object Broker, the Data Object Broker's connection attribute setting "commit level" must be set to 0. If not, the Gateway connection is rejected. Refer to Implementing Fail Safe Processing on page 38 for more information.
FSTABLENAME (FSTN)	Required only if FSLEVEL=1. Use to specify the name of the table to be used as a transaction table in the external DBMS that is denoted by the TRXDB parameter.

IDPREFIX (IDP)	Each pool of TIBCO Service Gateway for ODBCs must have a unique IDPREFIX (for example, `WINSK`). The prefix is used to construct a unique name for each Gateway. The ID prefix can contain up to five characters, to which the Gateway appends two hexadecimal digits and uses the result to log in to TIBCO Object Service Broker. It uniquely identifies each Gateway to TIBCO Object Service Broker. There is no default; you must specify a value.
LIBRARY (L)	The name of the local library for rules calls. Set the value to @SLK. Refer to the note at the end of this section for more information.
OTHERPASSWORD (OTPW)	Use to specify the password of the externally defined user ID. Has meaning only if EXTERNALUSERID=OTHER.
OTHERUSERID (OTUID)	The externally defined user ID. Has meaning only if EXTERNALUSERID=OTHER.
RECOVERYID (RECID)	Applicable only if Fail Safe Level=1. Identifies the authorization ID to use during recovery (that is, when querying the transaction table to see if the Gateway completed an in-doubt transaction). The ID can be up to eight characters. If not specified, the TIBCO Object Service Broker user ID (SESSIONUSERID) and password (SESSIONPASSWORD) are used.
RECOVERYPASSWORD (RECPW)	The password of the recovery user ID.
RESPONSEMODE	For use only with a Data Object Broker on z/OS. The mode that the Data Object Broker uses to release the Gateway at the end of a transaction. Use this mode to release a server more quickly for a new transaction in read-only situations. The default is ASYNC. Refer to the note at the end of this section for more information. Valid entries: ASYNC – The Data Object Broker releases the Gateway from the TIBCO Object Service Broker transaction upon sending the end of transaction request, provided no updates are requested. The next transaction can then be scheduled before the previous transaction is completed. ASYNC is recommended for online Gateways. SYNC – Causes the Data Object Broker to wait for the Gateway to complete end of transaction processing.
SCOPE	The length of time a connection to the external DBMS is held between the Gateway and an external DBMS. Valid values are TRANSACTION or SESSION, which maintain the connection for the life of a TIBCO Object Service Broker transaction or server session respectively. When the connection ends, the locks are released.

SEARCH	The library search environment for the rules to be executed by the Gateway. Set the value to L (for local). Refer to the note at the end of this section for more information.
SECLEVEL (SECL)	The level of authorization to use when accessing external DBMS data. The default is 0. Valid values: 0 – Access the external DBMS using the TIBCO Object Service Broker Gateway user ID and password as the connection ID and its password. 1 – Access the Gateway based on the user's TIBCO Object Service Broker session ID or group name, or externally defined user ID. Use an external security interface.
SERVERID (SID)	Identifies a pool of Gateways with common characteristics (for example, `SLKID1`). If you start up a pool of multiple Gateways with the same server ID, all members of the pool have the same values for all parameters.
SERVERTYPE (ST)	Identifies the type of Gateway. Specify SLK for TIBCO Service Gateway for ODBC. Specify ORS for TIBCO Service Gateway for Oracle.
SESSIONPASSWORD (SPWD)	Specifies the password of the TIBCO Object Service Broker user ID.
SESSIONUSERID (SUID)	Specifies a valid level-7 TIBCO Object Service Broker user ID (used internally by the Gateway to connect to TIBCO Object Service Broker).
TRXDB	Specifies the transaction database (data source for TIBCO Service Gateway for ODBC or Oracle SID for TIBCO Service Gateway for Oracle) that can be updated. All the SLK tables within a transaction used to update SLK data using the Gateway should make reference to this data source or Oracle SID. Required only if Fail Safe Level=1.
USERTYPE	The type of user that can connect to this server. When allocating a Gateway, the Data Object Broker first tries to allocate an online user to an online server. If it cannot, it allocates the user to a Gateway with USERTYPE=ANY. For use only with a Data Object Broker on z/OS. See the note at the end of this section for more information. Valid entries: ANY – Indicates that batch and online users can use the Gateway. This is the default. LOCAL – Reserves the server for user sessions executing in a Batch or CICS Execution Environment. Not valid for online users. ONLINE – Reserves the Gateway for online users only.

An ODBC or Oracle Gateway can be run with SEARCH=L, LIBRARY=@SLK settings or, alternatively, the INSTLIB parameter in mon.prm file can be set to @SLK and the search parameter for the Gateway set to “I” for “installation” in session.prm. The latter way is preferable, because it reduces the startup time for a Gateway.

The RESPONSEMODE and USERTYPE parameters have no effect on processing when the Data Object Broker runs on open systems. However, a z/OS Data Object Broker rejects a Gateway’s attempt to connect in case an attribute mismatch is detected. Refer to Startup Prerequisites on page 41.

See Also *TIBCO Object Service Broker Managing Security* about TIBCO Object Service Broker security.

Supplying the Gateway Configuration Parameters

Setting and Modifying the Gateway Configuration Parameters

The default Gateway configuration parameters are pre-set for all SLK Gateways during setup. If you want to specify individual values for a particular SERVERID, you use the @CONFIGURESERVER tool to set and modify gateway configuration parameters for the Gateway.

Creating the Gateway Configuration Parameters for a New SERVERID

To create or modify gateway configuration parameters for a SERVERID, complete the following steps:

1. Execute @CONFIGURESERVER from the workbench as follows:

   ```
   EX Execute Rule ==> @CONFIGURESERVER(SLK)
   ```

   A screen similar to the one below appears, showing a listing of defined SLK server IDs.

```
 Command==>                                                    SCROLL: P

       NUMBER   SERVERTYPE SERVERID
    ----------- ---------- --------
_             1 SLK        SLK01
_             2 SLK        SLKTEST
_             3 ORS        ORS01
_             4 ORS        ORSTEST2




A-ADD  D-DELETE S-SELECT
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

2. To delete server configuration parameters for a SERVERID, type D beside the respective entry.

```
To complete this command:
     NUMBER    SERVERTYPE SERVERID
   ----------- ---------- --------
 A           4 SLK        SLK01
Enter parameter(s)


 SERVERTYPE            ===>

 SERVERID              ===>









PFKEYS: ENTER=PROCESS 3=END 5=FIND NEXT 9=RECALL
```

3. To modify server configuration parameters for a SERVERID, type S beside the respective entry.
4. To add a new entry, type A beside one of the SERVERID entries listed on the screen and press Enter.
5. A screen similar to the one above appears, prompting you to enter a **SERVERTYPE** and **SERVERID**.
6. Type SLK in the SERVERTYPE field for TIBCO Service Gateway for ODBC or type ORS in the SERVERTYPE field for TIBCO Service Gateway for Oracle.
7. Type a value in the SERVERID field to create gateway configuration parameters and press Enter.

A screen similar to the one shown below appears, showing the default configuration settings.

```
          External Server Configuration Utility
COMMAND ==>
Server Type: ORS   Server ID: ORS01
                                                                Recommended/
Name              Value                                        Allowed Values
----------------- -------------------------------------------- ----------------
DEBUGLEVEL        0                                            0, 1
DUMP              N                                            Y, N
DUMPLIMIT         2048                                         '1...2147483647'
INTCOMMIT         Y                                            Y, N
KEEPLOG           N                                            Y, N
LOGMEDIA          TBL                                          TBL, SCR, PRT
SERVERSTATISTICS  N                                            Y, N
TRACE             N                                            Y, N



FCNKEYS: ENTER=VALIDATE 3=SAVE & EXIT 12=EXIT
```

8. Modify any default settings for the parameters in the Value field.

   Valid values are shown in the **Recommended/Allowed Values** field. For descriptions of the gateway configuration parameters, refer to Supplying the Gateway Configuration Parameters on page 33.

9. Press PF3 to save the settings and return to the workbench.

If you are modifying the setting for an active Gateway, you must recycle the Gateway for the new values to be applied.

The Gateway Configuration Parameters

The following configuration parameters are available for a Gateway:

DEBUGLEVEL	Used during problem analysis to determine which portion of the server code is in error. The default is 0. Valid values: 0 – Used during normal processing. 1 – The Gateway calls the interface, ignoring any replies.
DUMP	Specifies whether TIBCO Object Service Broker and ODBC or Oracle messages and control blocks should be logged (in the location specified via the LOGMEDIA configuration parameter). The default is N. Valid values: Y – Messages and control blocks are logged. N – No logging is in effect.
DUMPLIMIT	Specifies the amount (in bytes) of messages and control blocks that can be logged. The default is 2,048 bytes. Valid values are between 1 and 2,147,483,647 bytes.
INTCOMMIT	Specifies whether intermediate COMMITs are allowed. A COMMIT request is ignored in two cases: in a BROWSE transaction and when no updates against SLK tables are pending. A COMMIT request is considered intermediate if it is not ignored and is followed by access requests within the same transaction. The default is Y. Valid values: Y – Intermediate COMMITs are allowed. N – Intermediate COMMITs are not allowed.
KEEPLOG	Specifies whether the Gateway should keep the previous login startup. This applies only if the LOGMEDIA configuration parameter is set to TBL. The default is N. Valid values: Y – The Gateway log that matches the current SERVERTYPE, SERVERID, and SERVERUSERID is deleted on server startup. N – The Gateway log that matches the current SERVERTYPE, SERVERID, and SERVERUSERID is kept on startup. Subsequent logs are appended.

LOGMEDIA	Specifies where to store DUMP, TRACE, and error message information. The default is SCR. Valid values: PRT – DUMP, TRACE, and error message information is sent to the default print file for the TIBCO Object Service Broker user ID. SCR – DUMP, TRACE, and error message information appears in the session.log file. This can be viewed only after the server is shut down. TBL – DUMP, TRACE, and error message information is inserted into the @SERVERLOG TDS table (parameterized by SERVERTYPE, SERVERID, and SERVERUSERID). Browse this table while the Gateway is running. The SERVERUSERID is generated by the Gateway, and consists of the ID Prefix concatenated with two characters. For example, the Data Object Broker log displays the SERVERTYPE (SLK), SERVERID (SLKNT), and SERVERUSERID (SLKID01) in the S6BUA023I message as follows: `S6BUA023I SLK server SLKNT(SLKID01) logged on from '@SOY5C01'`
SERVERSTATISTICS	Controls how the Gateway transaction is managed. The default is N. Valid values: Y – The Gateway recycles its transaction at the end of the TIBCO Object Service Broker transaction. N – The Gateway runs as an endless TIBCO Object Service Broker transaction in BROWSE. **Note** For best performance do not change the default N, as the server transaction is independent of the client transaction.
TRACE	Specifies whether TIBCO Object Service Broker requests and SQL statements should be logged (in the location specified in the LOGMEDIA configuration parameter). The default is N. Valid values: Y – Both TIBCO Object Service Broker requests and SQL statements are logged. N – No logging is in effect.

Implementing Fail Safe Processing

You require Fail Safe processing if you need to guarantee data consistency when updating both TIBCO Object Service Broker TDS tables and external DBMS data in a single transaction.

Options for Fail Safe Processing

There are two Fail Safe processing levels available with the Gateway:

- At Fail Safe level-0, *no* Fail Safe processing is performed.
- At Fail Safe level-1, *one* external DBMS, denoted by an ODBC data source or Oracle SID, can be assigned as a transaction database so Fail Safe processing can be performed at the end of a transaction, if warranted by the nature of that transaction.

Fail Safe level-2 processing is not supported.

Requirements for Fail Safe Level-1 Processing

At Fail Safe level 1, the Gateways reject any update requests within a transaction if the data source (or Oracle SID) in the definition of the SLK table does not match the one specified as the transaction database for the Gateway. If, however, the request is accepted, a table of predefined structure for transaction data to be written (transaction table) is updated. This table's name is passed to the Gateway in the FSTABLENAME gateway parameter.

The FSTABLENAME parameter accepts no more than 16-character long values.

The placement of the Gateways' transaction tables should be made with care. The Gateway can ensure Fail Safe level-1 data integrity only to the external database containing the transaction table.

Fail Safe Processing and In-doubt Transactions

At the end of a transaction, the Data Object Broker requests that the Gateway commit outstanding updates. As part of commit processing, the Gateway updates the external transaction table to record the transaction end information. The Gateway then commits its work. If the Gateway does not respond to the Data Object Broker in a reasonable amount of time (or the connection to the Data Object Broker is lost), the transaction is flagged as in-doubt. Locks held on TDS data remain in place until the problem is resolved.

When a connection is re-established between the Data Object Broker and a Gateway with the same configuration as the one that failed, the Data Object Broker asks the Gateway if the in-doubt transaction completed. The Gateway checks the external transaction table to determine this. If the update completed in the external DBMS, the TIBCO Object Service Broker TDS updates are applied. If not, the updates are rolled back. In both cases, the locks are released.

Determining the State of the External Data

The table below shows how the Gateway reads the transaction database to determine the state of the external data.

If the external DBMS data is...	Then...
Updated	The Data Object Broker rolls forward the in-doubt TIBCO Object Service Broker transaction by committing TIBCO Object Service Broker data and releasing locks.
Not updated	The Data Object Broker rolls back the in-doubt TIBCO Object Service Broker transaction by discarding the intent list and releasing locks.

You can resolve in-doubt transactions only by starting a Gateway with exactly the same parameter settings as the Gateway in use at the time the transaction was placed in doubt.

Steps to Implement Fail Safe Processing

Step 1: Define an External Transaction Table

To implement Fail Safe level-1 processing, you must define an external DBMS table similar to the one shown below.

The sample Oracle table definition below shows Oracle terms, and that terms vary from one DBMS to another. The table in the example below holds a maximum of one record for each combination of a Gateway and Data Object Broker.

```
(dobid CHAR(8),
identsrv CHAR(8),
trxid NUMBER(10,0),
timestp1 NUMBER(10,0),
timestp2 NUMBER(10,0),
register VARCHAR (25))
```

Security Requirements

During Fail Safe level-1 processing, the Gateway inserts or replaces this table. If the Gateway is running with Security Level=1, this table is updated under the authorization of the TIBCO Object Service Broker client/session user ID. All users that need to access the external DBMS through the Gateway and run transactions that update both TDS and external data must have read and write access to this table.

Step 2: Define the Gateway Fail Safe Startup Parameters

Ensure that the gateway startup parameters listed below are defined as follows:

- Set FSLEVEL to 1.
- Set FSTABLENAME to the name of the external transaction table you defined in the external DBMS. This name is used “as is”, that is, no qualifier (catalog) and owner (schema) are prepended to it so it can contain those if needed, for instance “pubs.dbo.ostarfs.”
- Set TRXDB to the ODBC data source or Oracle SID denoting the database where you defined the transaction table that you plan to have participate in Fail Safe processing.

Startup Prerequisites

Prerequisites

Before you can start a Gateway, you must do one of the following:

- If the Data Object Broker is on z/OS, the Gateway must be identified to the Data Object Broker. To do this, define Gateway resources through the Data Object Broker's administration menu.
- Set up national language support for the Gateway, if necessary. Refer to *TIBCO Object Service Broker National Language Support* for setup and configuration information.

Default Resource Settings (z/OS only)

Use the Resource Management option available in the Administration control group of TIBCO Object Service Broker Administration Menu. Choose existing or add a new (PF5) pool of Gateways identified by type (SERVERTYPE) and group (SERVERID) to get to the Resource Detail screen. Use the Resource Detail screen to specify the connection attributes for the Gateway. The following table illustrates the attributes for a server with SERVERID=DEFAULT:

Group ID	INT RLBK	EARLY REL	LAST USER	COMMIT LEVEL	ONLINE ONLY
DEFAULT	N	N	N	0	N

Depending on the values you set for the gateway parameters (see Supplying the Gateway Startup Parameters on page 27), you must specify the connection attributes as follows:

The Gateway Parameter Settings	Connection Attribute Settings
SERVERTYPE=SLK	TYPE=SLK for TIBCO Service Gateway for ODBC.
SERVERTYPE=ORS	TYPE=ORS for TIBCO Service Gateway for Oracle.
SERVERID=<*serverid*>	Group ID=<*serverid*>
	INT RLBK must be set to N.

The Gateway Parameter Settings	Connection Attribute Settings
RESPONSEMODE=SYNC	EARLY REL=N
RESPONSEMODE=ASYNC	EARLY REL=Y
	LAST USER, can be set to Y or N.
FSLEVEL=1	COMMIT LEVEL=1
FSLEVEL=0	COMMIT LEVEL=0
USERTYPE=ONLINE	ONLINE ONLY=Y
USERTYPE=ANY	ONLINE ONLY=N
USERTYPE=LOCAL	ONLINE ONLY=Either Y or N

The LAST USER attribute set to Y makes the Data Object Broker attempt to allocate the Gateway to the same user at transaction switch time. This can be beneficial for the user if SCOPE is set to SESSION, as the external connection is to be reused.

The ONLINE ONLY attribute is associated with a schedule to be maintained separately. (PF10)

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information on defining resources and using the Administrator menu.

Starting the Gateway

Prerequisites for Startup

Before starting your Gateway, you must make sure that your Data Object Broker is running.

Understanding the Gateway Startup Process

At startup time, the Execution Environment receives several parameters, including one that instructs it to start up a number of Gateways. The first request for data from an external DBMS is made when a TIBCO Object Service Broker transaction begins. An ODBC or Oracle connection is established (if one is not already extant) to facilitate the transaction.

When the TIBCO Object Service Broker transaction ends, the ODBC or Oracle transaction ends as well. If the TIBCO Object Service Broker transaction completes successfully, the Gateway gets a synchronization message and commits the changes. If the TIBCO Object Service Broker transaction fails, the Gateway rolls back the changes. If a TIBCO Object Service Broker rollback statement is issued, the Gateway receives a separate message that causes it to roll back and then terminate the transaction at synchronization time.

The Gateway starts as part of the start up procedure of the Execution Environment that it is configured to run within. When you start the Gateway, it connects only to TIBCO Object Service Broker. Connection to the external DBMS occurs on the first data access request. Because a separate Execution Environment is not required for the Gateways, you can combine the gateway parameters with other Execution Environment parameters.

When a transaction is completed, the ODBC or Oracle connection ends or does not, depending on how the SCOPE parameter is designated in the Gateway configuration.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* and *TIBCO Object Service Broker for Open Systems Installing and Operating* for additional information on configuration and operations.

Shutting Down the Gateway

Shutdown Commands

The Gateway or Gateways automatically shut down when one of the following is shut down:

- The Execution Environment they are running within
- The <OS> monitor that spawned the Execution Environment
- The Data Object Broker to which they are attached

You can also shut down one or more Gateways using a STOPSERVER command. On z/OS, you issue the command **`MODIFY <DOBjobname>, STOPSERVER=<value>.`** On open systems, you issue the command **`hrncr STOPSERVER=<value>`**

<value> can be:

Value	Description
ALLSLK	Shut down all the SLK-type Gateways connected to this Data Object Broker.
ALLORS	Shut down all the ORS-type Gateways connected to this Data Object Broker.
srvidSLKS	Shut down all the Gateways that are members of the SLKS pool, that is, they have the SERVERID parameter set to SLKS.
srvidORS1	Shut down all the Gateways that are members of the ORS1 pool, that is, they have the SERVERID parameter set to ORS1.
SLK01	Stop the Gateway whose generated name (SERVERUSERID) is SLK01.

Chapter 4 Managing TIBCO Object Service Broker Data Definitions

This chapter provides information to manage TIBCO Object Service Broker external DBMS Data Definitions and to use the TIBCO Object Service Broker SLK table.

Topics

How to Access External DBMS Data

To access external DBMS data from TIBCO Object Service Broker, you must define a TIBCO Object Service Broker table of type SLK using the Table Definer. You also have the option of using the Table Definer to map a stored procedure for TIBCO Service Gateway for ODBC, that is, if you set SERVERTYPE to SLK. Using the Table Definer, you can access an existing table definition or create a new one.

Composition of an SLK Table

An SLK table consists of the following elements:

- Up to 4 columns as data parameters
- One or more columns as fields, of which one and up to 16 columns can constitute a composite primary key
- An optional location parameter
- Optional triggers

A stored procedure mapping can optionally have fields marked as:

- Return value
- Input parameters
- Input/output parameters
- Output parameters
- Result set columns

All these characteristics are considered known to the user at definition time.

Table Definition Requirements

Each SLK table must be unique; it must be defined to one and only one SERVERID. You can use the SERVERID parameter to identify a pool of Gateways that are considered equivalent by the Data Object Broker.

You indicate what type of Gateway your definition is meant for by specifying the SERVERTYPE parameter:

- SLK for TIBCO Service Gateway for ODBC
- ORS for TIBCO Service Gateway for Oracle

Accordingly, for TIBCO Service Gateway for ODBC, you must indicate which data source to use; for TIBCO Service Gateway for Oracle, you must indicate which Oracle SID to use.

Procedure to Define a Table

Complete the following tasks to define an SLK table:

1. Extract the external DBMS table definition, page 48
2. Invoke the Table Definer, page 49
3. Specify the server parameters, page 52
4. Select an external DBMS table, page 54
5. Define fields for the SLK table, page 59
6. Add control information, page 67

These tasks are described in Defining an SLK Table on page 48.

You can use the following optional tasks to assist in the management of the table:

- Dynamically change server parameters, page 70
- Extract SLK dictionary data, page 71
- Optimize access to the SLK table definition, page 73

These tasks are described in Post Table Definition Optional Tasks on page 70.

Defining an SLK Table

Task A Extract the external DBMS table definition

Overview

You can create an SLK table definition using one of two methods:

- Gateway method
- Extraction method

The Gateway method assumes that you are reading metadata from the external DBMS online. The extraction method can be used to retrieve pre-loaded metadata from a shadow dictionary in TIBCO Object Service Broker.

Using the Gateway Method

To use the Gateway method to define an SLK table, you *must* have a Gateway running. Specifying X (external) in the **Dictionary** field in the Table Definer indicates that the external DBMS (denoted by the data source or Oracle SID field) must be accessed by the Gateway (denoted by the **Serverid** field) to obtain table definition information.

Even if you run your Table Definer on z/OS, the respective Gateway must run on a supported system and attach to your z/OS Data Object Broker.

Using the Extraction Method

The extraction method can be used to pre-load SLK table definition information from external dictionaries and store it in TIBCO Object Service Broker TDS tables. You can then define SLK tables without having a Gateway running.

Specifying I (internal) in the **Dictionary** field means that you are able to view and map only to definitions contained in the shadow dictionary stored in TIBCO Object Service Broker.

If you use the extraction method, the data definitions are static; therefore, whenever changes are made to new or existing table definitions in the external DBMS, use the @SLKEXTRACT rule to load more up to date information. Refer to Task B, Extract SLK dictionary data, on page 71 for the steps required to reload the table definitions.

Task B Invoke the Table Definer

Overview

Invoke the Table Definer from the workbench using the DT define table option or the DT command. You can modify an existing table definition or define a new SLK table.

Accessing Existing Tables

To display the definition of an existing SLK table from the workbench, do one of the following:

- Type the name of an existing table beside the DT define table option and press Enter to display its definition.
- Type the name of an existing table in the command field as follows:

  DT SLKDEMO<Enter>
- Move the cursor to the DT define table option and press Enter. This displays the Object Manager screen, listing existing tables in your TIBCO Object Service Broker database. Scroll through this list to see which table you require. Use the **SELECT** command to filter the list to show only tables of type SLK:

  SELECT TYPE='SLK'

  To select a table, type **S** in the line command field and press Enter.

To define a new table:

1. Type the name of a new SLK table beside the DT define table option or type DT and the table name in the command field and press Enter.

   This displays a TDS table definition template.
2. Change the Type field at the top of the screen from TDS to SLK and press Enter.

   A default table definition screen appears, similar to the one shown below. If your screen display is set to 80 columns rather than 120, press PF11 to view the right-hand portion of the screen.

Default Table Definition Screen (left portion)

```
COMMAND==>                          TABLE DEFINITION

      Table: ODBCTBL          Type: SLK   Unit: HURON

   DBSource:                                          ServerType: SLK
   ServerId:                                       Proc/ProcResSet: N (P/R/N)
  TableName:                                          ServerOrders: Y (Y/N)
  Qualifier:                                       ImpliedUpdates: N (Y/N)
      Owner:                                           Dictionary: X (X/I)

  Location Parm    Default        Src Sourcename         Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                              _
   |--- External Field --------------|---- Metadata Field --------------------
    ExternalName                   Vrt Name             Typ Syn Len Dec Ord Req
    ------------------------------ -  ---------------  -  -- ----- --  -   -

  _
  _
  _
  _
  _
(P - parameter, K - key,  S - select, I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=TABLES 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

Default Table Definition Screen (right portion)

The following figure shows the right portion of the table definition screen, containing the additional **SQLDataType** and **Default** fields.

```
COMMAND==>                          TABLE DEFINITION

      Table: ODBCTBL          Type: SLK   Unit: HURON

   DBSource:                                          ServerType: SLK
   ServerId:                                  Proc/ProcResSet: N (P/R/N)
  TableName:                                     ServerOrders: Y (Y/N)
  Qualifier:                                   ImpliedUpdates: N (Y/N)
      Owner:                                       Dictionary: X (X/I)

  Location Parm    Default        Src Sourcename          Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                              _
   |--- External Field -------------- ----------------------------------------
    ExternalName                  Vrt  Default          SQLDataType
    ------------------------------ -  ---------------- ---------------
  _
  _
  _
  _
  _
(P - parameter, K - key,  S - select, I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=TABLES 5=COLUMNS 6=QUALIFIERS 9=OWNERS
Right edge of Window
```

Table Definition Screen Fields

Table	Displays the table name specified when you invoked the Table Definer. You can type in a new name to save the definition of the current table under a different name. Valid entries: A character string of up to 16 characters, beginning with a letter (A-Z) or a special character ($ or #), and continuing with more letters, special characters, digits (0-9), or underscore characters (_). A table name starting with an at (@) symbol denotes a table supplied by TIBCO Object Service Broker.
Type	Displays the table type SLK, which you changed in To define a new table: on page 49.
Unit	Displays the user unit associated with the table. You must use the Core screen to modify this field. Refer to Task F, Add control information, on page 67 for more information on the Core screen.

See Also *TIBCO Object Service Broker Managing Data* for more information on using the Table Definer.

Task C Specify the server parameters

Table Definition Screen Fields

These fields appear in the upper portion of the default table definition screen:

DBsource or Oracle SID	If SERVERTYPE is SLK, enter the name of the data source you want the Gateway to use to access the external DBMS. The ODBC data source is configured using the ODBC Administrator. For information on creating and managing ODBC data sources, refer to the relevant Microsoft documentation. If SERVERTYPE is ORS, enter the Oracle SID you want the Gateway to use to access the Oracle DBMS. Oracle SID is the symbolic name used by Oracle products to identify an Oracle database. **Note** This field is renamed to `Oracle SID` if you set the `Serverid` field to ORS. For information about Oracle SID, refer to the relevant Oracle documentation.
ServerType	Enter SLK for TIBCO Service Gateway for ODBC and ORS for TIBCO Service Gateway for Oracle. The default is SLK.

Serverid	You must specify the SERVERID of a Gateway to be associated with the table that you are defining. You can specify the SERVERID of a Gateway that is running (available) or you can specify the SERVERID of a Gateway that you plan to run during the data access session. A valid entry is an alphanumeric value up to eight characters long. You can view or use metadata information (qualifiers, owners, tables, and table columns) from the actual DBMS only if you specify the SERVERID of a running Gateway. If the SERVERID you specify does not denote a running Gateway, you can still define your SLK table manually. It is also possible to use extracted metadata to define an SLK table. Refer to the dictionary field (below) for information on specifying the dictionary parameter.
TableName or **Procedure**	Specify the name of the external table or procedure to which you are mapping your definition. **Note** This field is renamed to `Procedure` if you set the `Proc/ProcResSet` field to either P or R.
Qualifier	Enter the value of the qualifier (or catalog) that qualifies the table to which you are mapping. The qualifier can be entered directly into the field or selected from a list pressing PF6.
Owner	Enter the value of the owner (or schema) that qualifies the table to which you are mapping. The owner can be entered directly or selected from a list pressing PF9.
Proc/ProcResSet	Indicates if this definition maps a table/view, stored procedure or stored procedure result set:
	N — Specifies that the definition maps a regular table/view. (Default). For Oracle, this is the only valid value.
	P — Specifies that the definition being created/edited maps a stored procedure in the external DBMS. The name of the procedure in the external DBMS is required; specify the name in the **Procedure** field. Optionally, the stored procedure can create result sets (cursors) and/or result counts (numbers of rows affected by INSERT/UPDATE/DELETE requests). When you set the **Proc/ProcResSet** field to P, two auxiliary fields, @HANDLE@ and @RESULT@, are generated by the SLK table definer. Do *not* modify these field definitions. Refer to Adding Fields to a Stored Procedure Mapping on page 63 for additional field definition requirements.
	R — Specifies that the definition maps a stored procedure result set. You can define result set mappings that are usable for more than one stored procedure, because the name of the procedure is not required. The result set maps a pending cursor, that is, a temporary table that was built and can be fetched. When you set the **Proc/ProcResSet** field to R, an auxiliary data parameter, @HANDLE@, is generated by the SLK table definer. Do *not* modify this parameter definition. Refer to Adding Fields to a Stored Procedure Mapping on page 63 for additional field definition requirements.

ServerOrders	Specify Y if you want ordering performed by the server. If ordering must take place on the client side, specify N. Ordering is performed whenever an order clause is specified in a TIBCO Object Service Broker GET or FORALL request.	
ImpliedUpdates	Enter one of the following:	
	Y	In NOBROWSE transactions, a request including a GET or FORALL on the table is treated as an update request, so Fail Safe processing can be triggered if required at synchronization time. This is designed to be used primarily with stored procedures, and applies to regular tables as well.
	N	GET and FORALL requests at runtime are to be treated the regular way. (Default)
Dictionary	You can specify either an internal (I) or external (X) dictionary in the **Dictionary** field.	
	X	Specifies an external dictionary. If possible, metadata is read from the actual DBMS denoted by the data source. This is the only valid value for stored procedures.
	I	Specifies an internal dictionary. Metadata is read from the TDS tables populated by @SLKEXTRACT beforehand (parameterized by the data sources or Oracle SID).

Task D Select an external DBMS table

Overview

You can select an external DBMS table as the base for your SLK table. To select a table, type a valid external table name, and optionally a valid qualifier and a valid owner.

You can use asterisks (*) and question marks (?) as wildcards to search for external DBMS tables and owners.

Using PF Keys to Generate a List of Tables

If you do not know the table name, you can generate a list of tables using a valid PF key:

PF9	Generates a list of owners.
PF6	Generates a lists of qualifiers.

PF5	Generates a list of columns of a particular table.
PF4	Generates a list of tables defined in the external DBMS.

With the exception of PF4 (see Building a Table or Stored Procedure List below), the procedure is the same as Step 3: Extract the Desired Information on page 72.

See Also *TIBCO Object Service Broker Getting Started* for more information about wildcards and standard primary commands.

Building a Table or Stored Procedure List

Pressing PF4 in combination with the specified ODBC data source or Oracle SID produces an additional screen, shown in the following example, where you can specify a filter for the list of tables or stored procedures to be produced. Depending on whether your Proc/ProcResSet is set to R, P or N, a list of tables or stored procedures is produced. The example shows the external **TableName**, **Qualifier**, and **Owner** fields, which you could have already specified.

If the fields are left empty, all tables or procedures in the specified external database appear. To narrow the list of tables to be produced, specify whether to include the following table types (Y/N): **Table**, **System Table**, **View**, **Synonym**, or **Alias**.

Specifying Filters for Table Lists

```
  DBSource: ORACLE
ServerType: SLK              Table: ODBCTBL                    Proc/ProcResSet: N
  ServerId: SLK2          Location:                                 Dictionary: X

 Specify the Search Patterns for Table Retrieval in the target Database

TableName EM*                                          | Wildcards accepted, empty =*
Qualifier                                              | No wildcards accepted
    Owner Scott                                       | Wildcards accepted, empty = *
                                                       |-----------------------------
                                                       | Both Qualifier and Owner
                                                       | do preserve the case
Table Types (Y/N):
                TABLE Y
                 VIEW Y
         SYSTEM TABLE N
                ALIAS N
              SYNONYM N

<All existing types>  N


PFKEYS: 12=EXIT  Enter=SHOW TABLES
```

In this example, pressing Enter builds a list of tables satisfying the following criteria:

```
tablename LIKE ‘EM%’ & qualifier LIKE ‘%’ & owner ‘SCOTT’ & (type =
‘TABLE’ or TYPE = ‘VIEW’)
```

Sample List of Tables Returned

```
   DBSource: ORACLE
 ServerType: SLK                 Table: ODBCTBL               Proc/ProcResSet: N
   ServerId: SLK2              Location:                         Dictionary: X
                         Tables in the target Database
   TableName,  Qualifier, Owner                                    TableType
   ------------------------------------------------------------------
----------------
 - EM,, SCOTT                                                      TABLE
 - EM,, SCOTT                                                      TABLE
 - EMP_SNAPSHOT,, SCOTT                                            TABLE
```

```
  <S - select   E - expand>
PFKEYS: 12=EXIT  ENTER=EXPAND  3=SELECT AND MERGE WITH THE TABLE BEING DEFINED
```

Selecting the Table or Stored Procedure

After generating a table or stored-procedure list, you can:

- Type **E** beside the desired table or stored procedure and press Enter.

  All the column names for the selected external DBMS table or procedure are listed, as shown in the example below.

- Type **S** beside the table or stored procedure of your choice and press PF3.

  The fields for this table are merged with the fields of the SLK table being defined. When the table fields are merged, each external field either replaces the one with the same external name or is added to the list if there is no matching field.

  This action is identical to selecting the table columns when PF5 is used at the first table definition screen. This can be performed any number of times and the resulting list can contain fields of more than one table.

Column Names for Sample External DBMS Table

In the **ColumnName field** as shown in the following screen, column descriptions are produced for tables or views, procedure parameter descriptions are produced for stored procedures, and result set descriptions are produced for stored procedure result sets.

```
  DBSource: ORACLE
ServerType: SLK              Table: ODBCTBL              Proc/ProcResSet: N
  ServerId: SLK2          Location:                          Dictionary: X
 Tablename: CUSTOMER
 Qualifier:
     Owner: SCOTT


            Table Column Descriptions in the target Database
ColumnName                     Date Native Name       Precision  Length    Scale
------------------------------ type------------------ ---------- ---------- ---
ID                              3 NUMBER                 15         8
FIRST NAME                     12 VARCHAR2               30         30
LASTNAME                       12 VARCHAR2               30         30
USERNAME                       12 VARCHAR2               15         15
PASSWORD                       12 VARCHAR2               15         15
ADDRESS                         0 T_ADDRESS              1          1
STATUS                         12 VARCHAR2               8          8
EMAIL                          12 VARCHAR2               40         40



Data type is the ODBC-mapped SQL data type code  <0 means “unknown type”>
```

Valid PF Keys

PF Keys	Primary Command	Description
PF1	-	Displays corresponding help for the current field or screen.
PF2	**DOC**	Displays the Documentation screen, where you can document the table definition. Refer to Appendix A, Documenting TIBCO Object Service Broker SLK Tables, on page 97 for more information.
PF3	**SAVE**	Validates the definition information specified in the Table Definition screen. If the definition is valid, you are returned to the workbench.

PF Keys	Primary Command	Description
PF4	**TABLES** or **PROCEDURES**	Displays the screen used to specify the search values used to create a list of tables or stored procedures in the external DBMS. Press Enter to display the list.
PF5	**COLUMNS**	Displays the screen used to specify the search values for a list of: • Columns in the external table, in a table definition • Procedure parameters, in a stored procedure mapping where **Proc/ProcResSet=P** • Result set columns, in a stored procedure mapping where **Proc/ProcResSet=R** Press Enter to display the list.
PF6	**QUALIFIERS**	Displays a list of qualifiers for the specified data source. You can select one qualifier to be copied onto your definition screen.
PF9	**OWNERS**	Displays a list of owners for the specified data source. You can select one owner to be copied onto your definition screen.
PF13	**PRINT**	Prints the current definition.

Task E Define fields for the SLK table

Overview

A complete table definition or stored procedure mapping includes both the TIBCO Object Service Broker portion and the external DBMS portion, since the TIBCO Object Service Broker field definition is extended by the external DBMS column definition.

When you are finished defining the table or stored procedure, you can press PF3 to save the definition or PF12 to cancel the definition.

Table Definition Screen Fields for a Sample Table

```
COMMAND==>                              TABLE DEFINITION

    Table: ODBCTBL              Type: SLK     Unit: HURON

 DBSource: ORACLE                                        ServerType: SLK
 ServerID: SLK2                                  Proc/ProcResSet: N (P/R/N)
TableName: CUSTOMER                                 ServerOrders: Y (Y/N)
Qualifier:                                         ImpliedUpdates: N (Y/N)
    Owner: SCOTT                                       Dictionary: X (X/I)


  Location Parm    Default        Src Sourcename          Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                               _
   |--- External Field --------------|---- Metadata Field --------------------
    ExternalName                   Vrt Name              Typ Syn Len Dec Ord Req
    ------------------------------ -  ----------------  -   -- ----- --  -   -
S _ ENAME                             ENAME             S   V     10  0
S _ JOB                               JOB               S   V      9  0
S _ MGR                               MGR               Q   B      2  0
S _ HIREDATE                          HIREDATE          D   B      4  0
S _ DEPTNO                            DEPTNO            Q   B      2  0
S _ ID                                ID                Q   P      8  0
S _ FIRSTNAME                         FIRSTNAME         S   V     30  0
S _ LASTNAME                          LASTNAME          S   V     30  0
S _ ADDRESS.STREET                 _  ADDRESS           S   V     30  0
(P - parameter, K - key, S - select, I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=TABLES 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

Table Definition Screen Fields for a Sample Stored Procedure Mapping

```
COMMAND==>                              TABLE DEFINITION

    Table: SQLL              Type: SLK     Unit: DOC

 DBSource: sqls                                          ServerType: SLK
 ServerId: SLK2                                  Proc/ProcResSet: P (P/R/N)
Procedure: sstub;1                                  ServerOrders: Y (Y/N)
Qualifier: pubs                                  ImpliedUpdates: N (Y/N)
    Owner: dbo                                       Dictionary: X (X/I)

  Location Parm    Default        Src Sourcename          Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                                _
   |--- External Field -----------------|--- Metadata Field -----------------
    ExternalName                   Vrt Name            Typ Syn Len Dec Ord Req
    ------------------------------ -  ---------------- - -- ----- --  -   -
K _ @HANDLE@                          @HANDLE@         Q B      2  0
K _ @RESULT@                          @RESULT@         Q B      2  0
S _ RETURN_VALUE                    R  RETURN_VALUE     Q P      6  0
S _ @p1                             I  @P1              Q B      2  0
S _ @p2                             B  @P2              Q B      2  0
(S - select, K - key, I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=PROCS 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

When you set the **Proc/ProcResSet** field to P (procedure), two auxiliary fields, @HANDLE@ and @RESULT@, are generated. When you set the **Proc/ProcResSet** field to R (result set), an auxiliary parameter, @HANDLE@, is generated by the Table Definer. Do *not* modify these field definitions.

Selecting Fields

To select fields, type **P** if the field is to be a data parameter, type **K** if the field is to be a primary key, and **S** beside each field you want to select. By default, TIBCO

Object Service Broker includes the field in the definition if the selection is left blank. If you want to exclude a field, type **D** beside the field.

- To minimize processing overhead, select TIBCO Object Service Broker primary key fields that are the same as the primary key fields in the external DBMS. The number of fields you can select depends on the sum of all field lengths plus some control information. This sum must be less than or equal to 3915 bytes. For a detailed explanation of the formula used to calculate the total number of bytes of all fields, refer to *TIBCO Object Service Broker Programming in Rules*.
- TIBCO Object Service Broker holds up to 4 data parameters and up to 16 fields in a composite primary key; however, you can use the Table Editor and Table Browser only on tables with eight or less fields in a composite primary key.

Adding Fields to a Table Definition

You can add fields to your table definition using one of the valid PF keys listed, following the same procedures outlined in Task C, Specify the server parameters, on page 52 and Task D, Select an external DBMS table, on page 54.

To generate a list of columns from a table to which you can map your definition, press PF5. Before the list is generated an additional screen appears, where you can select a table by specifying a **Tablename**, **Qualifier**, or **Owner**.

You can also manually enter fields from the Table Definition screen. If an invalid value is entered, a message is returned.

Sample Screen to Extract Columns Based on Tablename and Owner

```
  DBSource: ORACLE
ServerType: SLK           Table: ODBCTBL                          Proc/ProcResSet: N
  ServerID: SLK2       Location:                                       Dictionary: X

Specify Table, Qualifier, Owner for Column Retrieval in the target Database

Tablename CUSTOMER                              | A non-empty value, no wildcards
Qualifier                                       | No wildcards accepted
    Owner SCOTT                                 | No wildcards accepted
                                                |--------------------------------
                                                | Both Qualifier and Owner
                                                | do preserve the case




PFKEYS: 12=EXIT  Enter=SHOW COLUMNS  3=MERGE WITH THE TABLE BEING DEFINED
```

Adding Fields to a Stored Procedure Mapping

Your field definition requirements are determined by whether your stored procedure mapping is for a stored procedure, **Proc/ProcResSet=P**, or a result set, **Proc/ProcResSet=R**.

Definitions for a Stored Procedure Mapping

The @HANDLE@ and @RESULT@ columns are generated and marked as primary keys. Your subsequent field definitions should map to the parameters of your stored procedure. All the procedure parameters must be mapped. The following types of parameters can be defined:

input	A value is passed to the procedure at invocation time or a default value is used. No value is returned by the procedure.
input/ output	A value is passed to the procedure at invocation time or a default value is used. A value is returned by the procedure.
output	A value is returned by the procedure. The optional return value falls into this category.

Use the **VRT** field to define the behavior of the parameters. Valid values are:

R	A return value (one per table definition). This field must follow the @RESULT@ field; if it does not, the definer moves the field to the third position in the definition.
I	An input parameter.
O	An output parameter.
B	An input/output parameter.

It is optional to define a return value. If the procedure returns a value and no field is mapped for it, the return value is discarded.

Definitions for a Result Set Mapping

The @HANDLE@ column is generated and marked as a data parameter. Your subsequent field definitions should map the fields of a procedure result set. The **VRT** field for all those fields is automatically set to S.

External Column Attributes

External Name	The name of the columns in the specified external table. This value can be entered manually or imported pressing PF4 or PF5. **Note** This name appears as specified and, possibly, in quotes (see the VRT attributes) in the SQL statement generated by the Gateway. Any string potentially acceptable at runtime can be entered. The Table Definition Screen Fields for a Sample Table on page 60 section illustrates that a user-type data member can be accessed in Oracle (via both ODBC and Oracle Gateways).
Data Type /Name	This value is returned by the Gateway when you merge fields. This value cannot be changed. The name refers to one of the unified data type names used to denote the variety of type names employed by DBMSs. **Note** The screen produced using one of the valid PF keys displays the true data type names returned by the actual DBMS, while the Table Definition screen displays the unified names. If the field in the external DBMS has configurable length (for example, CHAR) and scale, the length and scale are presented in parentheses after the data type name.

Vrt	Valid entries are:	
	V	Denotes that this field is virtual as far as the external DBMS is concerned. The Gateway never attempts to modify this field; however, it includes this field in the WHERE clause when it is appropriate. This helps to ensure better selectiveness when rows are updated or deleted. For example, if your external table does not have a primary key, or its primary key is not precisely mapped by your SLK definition (TIBCO Object Service Broker does not require or check this), you can use a virtual field to compensate for possible ambiguities, for example, ORACLE: rowid; Sybase: identity.
	Q	Denotes that the name of this field is to be enquoted whenever included in a SQL statement. This is necessary when field names contain special symbols.
	M	Denotes that this field is not to be included in INSERT column lists. You can then have Oracle object members as fields (for example, ADDRESS.STREET).
	W	Denotes that this field is to be used only in WHERE clauses.
	E	The expression provided as Default for this field is to be used as the value in INSERT/UPDATE statements (for example, SEQ.NEXTVAL).
	1	The combination of Q and V.
	2	The combination of Q and W.
	3	The combination of Q and E.
	4	The combination of M and E.
	R	Denotes a return value column of a stored procedure mapping.
	I	Denotes an input parameter of a stored procedure mapping.
	O	Denotes an output parameter of a stored procedure mapping.
	B	Denotes an input/output parameter of a stored procedure mapping.
	S	Denotes a result set column of a stored procedure result set mapping.
	blank	None of the above. (Default)

If you use the merging facility, the SLK Gateway always places the values it suggests that you use in the **SYN** fields. You can change the suggestions, keeping in mind that not all data types can be coerced into each other.

Metadata Fields

Name	The TIBCO Object Service Broker name for the column. This name must be unique within the SLK table. You can give a field the same name as a field in another table. If you are moving data between two tables, giving the fields the same name simplifies the process. Valid values include a character string of up to 16 characters beginning with a letter (A-Z) or a special character ($ or #), and continuing with more letters, special characters, digits (0-9), or underscore characters (_). A table name starting with an at (@) symbol denotes a TIBCO Object Service Broker supplied table.
Typ	Specifies the TIBCO Object Service Broker semantic data type of the column. You can use any valid TIBCO Object Service Broker semantic type, provided that the semantic type and syntax combination is valid. Valid combinations are described in *TIBCO Object Service Broker Programming in Rules*. Valid entries:
	C COUNT
	D DATE
	I IDENTIFIER
	L LOGICAL
	Q QUANTITY
	S STRING
Syn	Specifies the TIBCO Object Service Broker syntax of the column. You can use any valid TIBCO Object Service Broker syntax type, provided that the semantic type and syntax combination is valid. Valid combinations are described in *TIBCO Object Service Broker Programming in Rules*.
Len	Specifies the TIBCO Object Service Broker length of the SLK column, in bytes.
Dec	For syntax P only. Specifies the number of digits to the right of the decimal point. The number of decimal places must be smaller than twice the length of the entire field.

Req	Specifies if the field is required. Any field can be a required field. A primary key field is required by default. Valid entries:	
	Y	Required. Every occurrence in the table must have a value or default for this field. Inserting or editing an occurrence without valid values in required fields causes an exception to be raised.
	N	Not required.
	blank	Not required.
Ord	Specifies that data retrieved from this table is sorted on this field. Valid entries:	
	A	Ascending sequence.
	D	Descending sequence.
Default	Specifies a default value to be assigned to the field if the field is null at insert time.	

A wrong TIBCO Object Service Broker syntax can cause a conversion error, since the Gateway must convert each field of each row to the syntax as defined in the SLK Table Definition.

Data Type Translation

Refer to *TIBCO Object Service Broker Programming in Rules* for an explanation of TIBCO Object Service Broker semantic types and syntax.

Task F Add control information

Location Parameter Information

You can define a location parameter for SLK tables. A location parameter is required only if you need to access an external DBMS through a peer server and external Gateway, both associated with a different Data Object Broker (remote node).

If you always access the SLK table remotely, the node from which you request the access can have either a minimal or a full definition.

Minimal Definition

A minimal table definition consists of the following elements:

- The table name, which must be the same at both locations
- The location parameter

  The name of the remote node where the full definition is located *must* be supplied through the use of the **Default** field, **Src** field, or **Src** and **Sourcename** field.

A minimal definition with a location parameter means you always access data at a remote node. The table type for such a minimal definition must be TDS.

Full Definition

A full table definition that has a location parameter indicates that you can access data at either the local node or at a remote node.

The table type specified in either a full or minimal definition does not have to match the table type of the full definition at the remote node. For example, a full definition of type TDS used to access TDS data on the local node can also be used to access an SLK table with the same name at a remote node.

Specifying Event Rule Information

Use the event rule segment of the Table Definition screen to provide additional controls over access to a particular table. Define event rules based on the type of access performed.

Event rules are always called when the table is accessed according to the access type specified. All rules that apply to a specific access type are executed in the order they appear in the event rule section. They cannot access tables on a remote node.

Types of Event Rules

Three types of event rules can be defined:

Validation	Verifies the value of an occurrence when the table is being modified (such as checking the validity of a field value).
Trigger	Causes additional processing to take place when a table is accessed. For example, a trigger rule can be used to create an audit trail or update other tables.
ProcEvnt	Processes stored procedure results.

Event Rule Fields

The event rule information is entered in the scrollable event rule section. To define event rules, complete the following fields:

Type	Specify the type of event rule. Valid entries:		
	V Validation Rule No database updates are allowed during the validation process. The rule must be a function that returns Y (yes), the validation is successful, N (no), the validation is not successful, or a message explaining why it is not successful.		
	T Trigger Rule A trigger rule cannot be a function or change the contents of the triggering row, and cannot use the TRANSFERCALL statement. Nested triggers are permitted.		
	P Stored Procedure Result Processing Rule The rule specified in this entry is invoked to process the result set of the stored procedure (for TIBCO Service Gateway for ODBC only). Refer to Coding Event Rules on page 90 for more information about event rules for stored procedures.		
Access	Specify the type of access or manipulation to be performed on the data, causing the event to be executed. Valid entries:		
	Validation Rules	**Trigger Rules**	**ProcEvnt Rules**
	W - Any write (insert, replace, delete)	W- Any write (insert, replace, delete)	N - Entry disabled
	I - Only insert	I - Only insert	O - Row count returned
	R - Only replace	R - Only replace	U - Cursor (result set) returned
	D - Only delete	D - Only delete	
		G - Any retrieval	
Rule	Specify the name of the event rule.		

See Also *TIBCO Object Service Broker Managing Data* for more information on location parameters, event rules, and minimal table definitions.

Post Table Definition Optional Tasks

Task A Dynamically change server parameters

Tools for Dynamic Modification

At runtime, you can dynamically modify the server parameters using the tools SETXPARM and RESETXPARM. The use of these tools reduces the number of table definitions required to map the external data. The changes to the server parameters are stored in either of the following two session tables:

@SRVRPRMS_TYP	Manages global changes to the table type.
@SRVRPRMS_TBL	Manages specific changes to an individual table. The changes are in effect for the duration of the transaction or until SETXPARM is invoked again or the overrides are reset. The changes apply to definitions at locations specified as arguments to SETXPARM and RESETXPARM.

The Server Parameters That Can Be Overridden at Runtime

The following server parameters can be changed dynamically with SETXPARM and RESETXPARM:

- SERVERTYPE
- SERVERID
- DBSOURCE
- QUALIFIER
- OWNER
- TABLENAME
- SERVERORDERS
- IMPLIEDUPD
- CODEPAGE
- VERSION

Examples Using SETXPARM and RESETXPARM

- This example sets the SERVERID for all SLK tables to TORONTO:

```
CALL SETXPARM('TABLETYPE', 'SLK', 'SERVERID', 'TORONTO');
```

- This example resets the SERVERID for SLK tables to the Table Definer default value:

```
CALL RESETXPARM ('TABLETYPE', 'SLK', 'SERVERID', '');
```

See Also *TIBCO Object Service Broker Shareable Tools* for detailed descriptions of the SETXPARM and RESETXPARM tools.

Task B Extract SLK dictionary data

Step 1: Execute the @SLKEXTRACT Rule

To begin copying dictionary data from an external DBMS into the shadow dictionary, run the @SLKEXTRACT rule from the workbench. The @SLKEXTRACT rule produces a screen similar to the one shown below:

```
             External Server Metadata Extract/Clean Utility
Please enter
EITHER
   Serverid                                    | The ID of a running SLK server
   DBSource                                    | A preconfigured ODBC data
                                               | source

OR
 Tablename                                     | An existing table of type SLK
                                               | to reference both of the above
ALSO
If your SLK server is running remotely, please enter
   location                                    | A DOB node where
                                               | your server is running
-------------------------------------------------------------------------------
Search pattern (TABLES and COLUMNS only)
 Tablename                                     | These refer to the tables
 Qualifier                                     | in the external DBMS
     Owner                                     |
Table types (TABLES only) : TABLE  Y      VIEW Y   SYSTEM TABLE N
                            ALIAS  N   SYNONYM N

PFKEYS:  3,12=EXIT    Extract:  4=tables   5=columns   6=qualifiers   9=owners
      Clean up the dictionary: 16=tables  17=columns  18=qualifiers  21=owners
```

Step 2: Enter the Serverid, DBSource, and Location

In the displayed screen, enter the **Serverid**, **DBSource**, and **Location** and press Enter.

If you already have an SLK table defined with the Serverid and DBSource jointly denoting the external DBMS from which you want to extract metadata, enter the name in the **TableName** field and press Enter. The **Serverid**, **DBSource**, **Qualifier,** and **Owner** fields are filled in automatically.

If the SLK table definition resides on a remote node and you enter this location, the values for **Serverid**, **DBSource**, **Qualifier**, and **Owner** are extracted from the definition at that location.

Step 3: Extract the Desired Information

Use the PF keys listed below to extract the desired information. After making a selection, you are prompted for confirmation.

PF4	Extracts tables.
PF5	Extracts columns.
PF6	Extracts qualifiers (if supported).
PF9	Extracts owners.

Filters

If extracting columns and/or tables, you can restrict the amount of metadata extracted by using the filters in the lower portion of the screen. The following filters can be used with both tables (PF4) and columns (PF5):

TableName	Enter a search pattern with optional wildcards.
Qualifier	Enter a value (if left empty, no qualifier is applied).
Owner	Enter a search pattern with optional wildcards.

If selecting tables only (PF4), you can also specify Y or N in each of the five fields (**Table**, **View**, **System Table**, **Alias**, and **Synonym**). Specifying N means that dictionary information for tables of this type is not extracted.

To extract tables and columns metadata for a particular qualifier, you must use PF4 and/or PF5 for each value.

Step 4: Clean Up the Shadow Dictionary

After extracting the desired information, use the PF keys listed below to clean up the shadow dictionary. After making a selection, you are prompted for confirmation.

PF16	Cleans up tables.
PF17	Cleans up columns.
PF18	Cleans up qualifiers (if supported).
PF21	Cleans up owners.

Task C Optimize access to the SLK table definition

Binding the Table Definition

To optimize access to the SLK table, you can bind an SLK table definition; you *cannot* bind its data. SLK tables for which you request binding are bound to both the Execution Environment and the Gateway when they are accessed from a rule.

You can dynamically request a re-bind for the Execution Environment if you want to change a bound definition. To do this, modify the definition using the Table Definer and run the $BINDOBJECT routine against the new definition. Subsequent access requests against this definition, issued within the Execution Environment where you run $BINDOBJECT, refresh the definition in the Gateway.

See Also *TIBCO Object Service Broker Application Administration* for information on binding tables and on the $BINDOBJECT routine.

Chapter 5 Processing External DBMS Data

This chapter shows you how to access the TIBCO Object Service Broker SLK tables.

Topics

Using the Table Browser or the Table Editor to Access Tables

When you access external DBMS data from TIBCO Object Service Broker, it translates the request into an external (ODBC or Oracle) API call and external DBMS field data types are translated to the TIBCO Object Service Broker field types defined in the SLK table. To access the SLK tables for external data access, you can use various workbench tools, such as the Table Browser and the Table Editor, or you can use the rules language.

TIBCO Object Service Broker Requests vs. SQL Requests

When a TIBCO Object Service Broker request for SLK data is first processed by the Gateway, TIBCO Object Service Broker generates an SQL statement. This statement contains references to table columns as outlined in the SLK definition. As some SQL-specific rules apply, the table definition contains the Vrt field (refer to Vrt on page 65) to provide the issuer of the request with some control over this process.

The names of the columns in the SQL statement appear exactly as specified in the external name field of the SLK definition, possibly enquoted. This means that any expression that makes sense for the target DBMS can be specified. You use the Vrt field as follows:

Meaning of the SQL Column	Value of Vrt	Comments
Virtual	V or 1	The column must not be updated, that is, its name must appear neither in the SET clause of an UPDATE statement nor in the column list of an INSERT statement.
Quoted	Q, 1, 2, 3, or 4, as required	The nature of the column name is such that it must be enquoted whenever used (for example, “DELETE”, “Number of items”).
Member of class	M or 4	The column must not appear in the column list of an INSERT statement (for example, member of Oracle's object such as address.street).

Meaning of the SQL Column	Value of Vrt	Comments
Where only [a]	W or 2	The column must appear only in the WHERE clause (for example, an expression with side effects). The value of the column in any row retrieved is NULL.
Expression as default	E, 3, or 4 as required	The column's default value must by used “as is” in the VALUES clause of an INSERT and the SET clause of an UPDATE statement (for example, if an SLK definition maps an Oracle table and SEQ is an Oracle sequence, SEQ.NEXTVAL specified as default for a column in that definition results in behavior similar to TIBCO Object Service Broker's IDgen).
Virtual and Quoted	1	
Quoted and Where only	2	
Quoted and Expression as default	3	
Member of class and Expression as default	4	

a. When using Where only in complex WHERE clauses, TIBCO Object Service Broker can transform the original logical expression into an equivalent DNF (disjunctional-normal form) with a different number of column references. For example, the expression (R1 & (R2 | R3)) is submitted to the target DBMS as (R1 & R2) | (R1 & R3), where R1, R2, and R3 are logical terms <field><relational-operator><value>. Also, if <value> in a term is an expression, the term is excluded from the WHERE clause passed to the target DBMS, so that TIBCO Object Service Broker, as opposed to the target, carries out the evaluation of the rows upon return from the target DBMS.

Oracle requires that a table correlation name precede an object member notation. To enable the use of such notation, both TIBCO Service Gateway for Oracle and TIBCO Service Gateway for ODBC (when connected to Oracle) assign “T” as correlation name to the table reference in SELECT and UPDATE statements they generate. Accordingly, the external name of the respective field in the SLK definition is expected to begin with a “T.”, for example, T.address.street.

Browsing or Editing a Table

Use the Table Browser to browse a defined SLK table by typing the table name next to the BR browse table option and pressing Enter. To edit a table, type the table name next to the ED edit table option and press Enter. In both cases a screen similar to the following appears.

Sample SLK Table with Data Presented in TIBCO Object Service Broker Table Format

```
BROWSING TABLE   :  SLKDEMO
COMMAND ==>
                                                                SCROLL: P
        ABC      CDE
_ ------------ -----
_          100 1000
_          101 1000
_            3 BB00
_           77 770
_            5 vvv0
_           14 140
_           55 55500
_           99 990
_
_
_
_
_
_
_
_
_
PFKEYS: 1=HELP 5=FIND NEXT 9=RECALL 18=EXCLUDE 13=PRINT 3=END 14=EXPAND
```

Using the Table Browser or the Table Editor, you browse and edit an SLK table in the same way you would browse or edit any other TIBCO Object Service Broker table.

If your table definition contains fields longer than 260 bytes, you must do one of two things: use the Single Occurrence Editor (SOE) from the Table Editor to edit them and use **`SELECT LIKE`** to retrieve occurrences based on the values of these fields.

See Also

- *TIBCO Object Service Broker Application Administion* for more information about administering a distributed processing environment.
- *TIBCO Object Service Broker Managing Data* for more information about minimal definitions, and browsing and editing tables in TIBCO Object Service Broker.

Using Rules to Access Tables

Accessing external DBMS data using the rules language is similar to accessing TIBCO Object Service Broker data. The following sections outline the specifics encountered while using rules, and also point out normal TIBCO Object Service Broker behavior that you must consider when building applications.

Use the following sections in conjunction with *TIBCO Object Service Broker Programming in Rules*, which describes TIBCO Object Service Broker rules language statements and coding.

Retrieval Processing

A single cursor is used to retrieve data from the external DBMS for each retrieval statement (GET or FORALL) in your rule.

- If the BROWSE flag is set to Y, no locks are taken. If the BROWSE flag is set to N, locks are taken on all external tables accessed via the FORALL statement. Also, if a subview against an SLK table is used, locks are taken according to the behavior warranted by the subview's lock mode setting.
- If the server parameter SERVERORDERS=N, a GET ...ORDERED statement retrieves all data that satisfies the selection criteria and sorts it in the Execution Environment before returning the first occurrence that meets the selection criteria. If the server parameter SERVERORDERS=Y, the ORDERED clause is passed to the external DBMS.

GET Statement

A GET statement retrieves the first occurrence in the SLK table that satisfies the specified selection criteria.

FORALL Statement

A FORALL statement is a looping construct that processes a set of occurrences. The body of the loop consists of the statements to be executed for each occurrence satisfying the selection criteria. FORALL statements can be nested provided that they refer to different TIBCO Object Service Broker table parameter instances.

Occurrences are returned to TIBCO Object Service Broker in the order in which the Gateway passes them. If you require a different order, you must include an ORDERED clause in your FORALL statement or mark the relevant fields via the Table Definer.

Replace (Update) Processing

For the TIBCO Object Service Broker occurrence being replaced, all parameters and primary keys, with the AND logical operator between them, are used to make up the criterion describing the rows to replace in the external DBMS.

If the SLK table is defined with only data parameters and/or fields that result in an empty SET clause in a REPLACE statement, that REPLACE does not occur and an error message is produced.

Insert Processing

At insert time, all parameters and fields of the SLK definition except for those marked with VRT=V, W, M, 1, 2, or 4 are part of the column list of the INSERT statements submitted to the external DBMS.

Delete Processing

This is the same as Replace (Update).

Transaction Streams and Gateways

If you issue a TIBCO Object Service Broker EXECUTE statement within a parent transaction, a child transaction stream is created. The number of streams allowed in a TIBCO Object Service Broker transaction depends on the TRANMAXNUM Execution Environment parameter. Each transaction stream accessing SLK data requires its own gateway.

Ensure your system administrator is aware of the number of gateways required to accommodate all transaction streams accessing data in a single transaction.

Using TRANSFERCALL or DISPLAY & TRANSFERCALL statements in a rule minimizes the number of Gateways an application could require and reduces the possibility of locking contentions in the external DBMS.

See Also *TIBCO Object Service Broker Parameters* for your operating environment for information about the TRANMAXNUM Execution Environment parameter.

Synchronization and Recovery

Transaction Synchronization

A transaction in the external DBMS spans the same length of time as a TIBCO Object Service Broker transaction. The BROWSE/NOBROWSE flag on the workbench determines the locks taken on external data.

A COMMIT causes the external DBMS to release all locks and close all cursors. COMMITs are issued by the TIBCO Object Service Broker Gateway at the end of a transaction even if no updates are made so locks taken are released.

Intermediate COMMITs in NOBROWSE transactions, while any updates to SLK tables are pending, could be allowed depending on the setting of the INTCOMMIT configuration parameter (refer to INTCOMMIT on page 36).

- The Oracle Gateway preserves all open cursors beyond a COMMIT point. The ODBC Gateway preserves, as well, all open cursors when connected to Oracle, MS SQL Server, and DB2 UDB. Cursor stability is not guaranteed for other possible data sources.
- An intermediate COMMIT, if propagated into the target DBMS, can compromise transactional integrity and, therefore, should be used with care.
- Normally, all locks acquired prior to a COMMIT are released by the target DBMS (except those used implicitly by the cursors currently open); accordingly, a TIBCO Object Service Broker transaction should not rely on locks between commit points and on the durability of updates throughout its lifetime.

Synchronization Results

If updates are performed, the following results occur for the following requests:

Intermediate COMMIT	The Gateway issues COMMIT. Subsequent requests for data are considered erroneous.
Intermediate ROLLBACK	The Gateway issues ROLLBACK. Subsequent requests for data are considered erroneous.

End of Transaction ROLLBACK	The Gateway issues ROLLBACK.
End of Transaction Synchronization	The Gateway issues COMMIT.

If no updates are performed, the following requests have these results:

Intermediate COMMIT	The Gateway does not receive COMMIT if there is no update.
Intermediate ROLLBACK	The Gateway does not receive ROLLBACK if there is no update.
End of Transaction ROLLBACK	The Gateway issues ROLLBACK.
End of Transaction Synchronization	The Gateway issues COMMIT.

The exception COMMITLIMIT does not apply to SLK tables. Requests to update external data are processed as they are encountered and are not buffered in the intent list.

Transaction Recovery

Updating External Data Only

Transactions that update only external data are recoverable under the external DBMS.

Updating TIBCO Object Service Broker and External Data

Fail Safe level-1 processing provides a method of ensuring data integrity when a TIBCO Object Service Broker transaction updates both external and TIBCO Object Service Broker data in the same transaction. If you do not request Fail Safe processing, transactions that update both external and TIBCO Object Service Broker data can result in discrepancies if the Gateways or the Data Object Broker abnormally terminates during transaction end processing. Refer to Implementing Fail Safe Processing on page 38 for more information.

Recovering from an Abnormal Data Object Broker Shutdown

The Gateways do not automatically log in to a Data Object Broker that is reactivated after an abnormal Data Object Broker shutdown.

Error Handling and Recovery

The TIBCO Object Service Broker runtime environment signals system exceptions, permitting an application to recover from an error. A three-level hierarchy of exceptions exists. The ERROR exception is the top of the hierarchy and is intended to be a catchall exception. Each exception traps the exceptions that appear below it in the hierarchy. All errors encountered when accessing external data through the Gateways are trapped under one of the TIBCO Object Service Broker exceptions listed below.

ERROR

An error is detected and no lower-level exception handler exists in the application.

ACCESSFAIL

An error is detected and no lower-level exception handler exists in the application.

GETFAIL	No occurrence satisfies the selection criteria.
DELETEFAIL	The primary key specified for a DELETE statement does not exist (no rows deleted).
INSERTFAIL	The primary key provided for an INSERT statement already exists.
REPLACEFAIL	The primary key provided for a REPLACE statement does not exist (no rows replaced).

INTEGRITYFAIL

An attempt to violate data integrity is detected and no lower-level exception handler exists in the application.

LOCKFAIL	There is a lock on an occurrence of a table or a table is unavailable.

SECURITYFAIL	Permission for the requested action on the TIBCO Object Service Broker object is denied. This occurs if the connection ID does not have permission to perform the requested action on the specified object in TIBCO Object Service Broker. This can also occur if SECLEVEL=1 and the EXTERNALUSERID parameter is set to GROUP and the group name is greater than eight characters long.
SERVERBUSY	A new transaction requested a Gateway and no Gateway is available. Control is passed back to the rule, so that the rule can try the transaction again. If this exception is raised too often, consider requesting more Gateways or reviewing the amount of work being done in your transactions.
SERVERERROR	The Gateways made a request to the external DBMS and got an error code that does not map to one of the specific TIBCO Object Service Broker exceptions. The ON SERVERERROR handler can call @SERVERERROR to parse the error message (contained in ENDMSG). You must pass @SERVERERROR the contents of RETURN_MESSAGE, which is in the following format: *ppp*SL*nnnx serverid serveruserid: Message*

ppp	The user-specified 3 character product ID.
nnn	The external DBMS message number.
x	The type of message (E for error, W for warning, and I for information).
serverid	The server ID of the Gateway.
serveruserid	The server user ID (IDPREFIX + ###) of the the Gateway.
Message	The actual error message.

SERVERFAIL	A transaction is in progress when the connection to a Gateway breaks or the Gateway fails. Control is passed back to the rule for transaction cleanup.

See Also

- *TIBCO Object Service Broker Programming in Rules* for more information.
- *TIBCO Object Service Broker Shareable Tools* for more information on the @SERVERERROR and RETURN_MESSAGE tools.

Debugging Applications

Using the TIBCO Object Service Broker Rule Debugger

You can use the Rule Debugger to identify and fix errors within your applications. You can also make and test certain ad hoc changes to your rules. The Debugger stops the rule execution at events that you specified from within the Debug screen.

Using the Gateway Trace

To see the SQL statements being sent to the Gateway by your rules, turn the TRACE configuration parameter on using the @CONFIGURESERVER tool. The SQL statements for all SLK table accesses are included in the trace. Refer to Supplying the Gateway Configuration Parameters on page 33 for more information on @CONFIGURESERVER. Use the SERVERID parameter to view only SQL statements created by your rules. A dedicated server is required.

Reporting Problems with the Gateways

Refer to How to Contact TIBCO Support on page xvi for information about reporting problems to TIBCO Support.

Have the following information available when reporting server-related problems to TIBCO Support:

- SLK table definition(s) and sample data
- Listings of all server control tables for the server
- The server session log and control cards
- Output from the server dump and/or trace (if applicable)

See Also *TIBCO Object Service Broker Programming in Rules* for more information on debugging rules.

Chapter 6 Invoking Stored Procedures with TIBCO Service Gateway for ODBC

This chapter provides information to invoke Stored procedures with TIBCO Service Gateway for ODBC.

Topics

Stored Procedure Invocation

You can invoke a stored procedure from within your rules, using CALL or EXECUTE or TRANSFERCALL statements if it is procedural, or depending on its definition you can code it as a function. Your stored procedure can execute within a transaction or start a new transaction. Optionally, you can also invoke event rules to be run with a stored procedure.

Please note the following:

- Stored procedures are not supported by the Service Gateway for Oracle.
- Stored procedures are not supported by the z/OS EE.

Invoking a Stored Procedure

If your stored procedure is defined with a return value field, that is, a field defined with VRT=R, you can invoke it as either a function or a procedure from within your rules. If there is no return value field, you can invoke it only as a procedure. Upon completion of its execution, the fields of the stored procedure mapping, including the return value, are set to the values of the parameters as returned by the procedure, while input parameters stay intact.

Usage

You can use the following syntax within your rules to invoke a stored procedure:

Positional Invocation

```
CALL <stored_procedure>(<parameter1-value>,<parameter2-value>,…)

EXECUTE
<stored_procedure>(<parameter1-value>,<parameter2-value>,…)

TRANSFERCALL
<stored_procedure>(<parameter1-value>,<parameter2-value>,…)
```

where

<stored_procedure>	The TIBCO Object Service Broker name of the stored procedure mapping.

<parameterN-value>	The value to be passed for the parameter # N, counting input and input/output parameters only. If you specify NULL the parameter is passed as NULL; if you specify an empty string (''), the default, if any, is used for the parameter; all input and input/output parameters preceding the last output parameter must be specified; an unspecified parameter is considered an empty string.

Keyword Invocation

```
CALL <stored_procedure> WITH
<parameter1>=<parameter1-value>&<parameter1>=<parameter1-value>&…

EXECUTE <stored_procedure> WITH
<parameter1>=<parameter1-value>&<parameter1>=<parameter1-value>&…

TRANSFERCALL <stored_procedure> WITH
<parameter1>=<parameter1-value>&<parameter1>=<parameter1-value>&…
```

<stored_procedure>	The TIBCO Object Service Broker name of the stored procedure mapping.
<parameterN>	The TIBCO Object Service Broker name of the field representing the parameter. There is no need to specify all parameters; any unspecified parameter is considered an empty string.

Functional Invocation

```
<stored_procedure>(<parameter1-value>,<parameter2-value>,…)
```

Functional invocation is applicable only for stored procedures with a return value, that is, a field with VRT = R.

Starting a New Transaction

Using EXECUTE or TRANSFERCALL statements, you can invoke your stored procedures as separate TIBCO Object Service Broker transactions; no functional invocation is possible using these statements. However, if a return value is set by the procedure you can use the GETENDMSG tool to retrieve the return value, if any, set by the procedure. Use the same syntax as described in the CALL statement above when using EXECUTE or TRANSFERCALL statements.

See Also *TIBCO Object Service Broker Shareable Tools* about the GETENDMSG tool.

Coding Event Rules

Using the event rule feature, you can associate business rules and policies with your stored procedure mapping. Based on defined accesses, these rules are run whenever data in your SLK table is manipulated. You can code ProcEvnt event rules (Type=P) for stored procedures.

Coding Considerations

Consider the following when coding your event rules:

- Some DBMSs do not notify the caller when an event activated by a row count returned (update, insert, or delete) occurs or they require special configuration for this kind of notification. If no notification is carried out by the target DBMS, these event rules are not called.
- The event rules are executed as part of the transaction that spans the stored procedure execution. They cannot change the flow of control within the procedure.
- For most target DBMSs, the event rule is not aware of whether the respective event was fired after the actual procedure completed or during its execution. The ODBC standard does not prohibit these events from being fired at procedure runtime so, theoretically, the parameters can contain relevant intermediate values. To provide for this case, TIBCO Service Gateway for ODBC makes available to the caller the current parameter values as set by the procedure.
- Event rules fired off by a cursor returned, fetch the currently pending result set by means of a FORALL or GET statement against the TIBCO Object Service Broker mapping of this result set. If none is issued, the result set is considered discarded; if a GET is issued, a row, if any, is returned and the result set is discarded. The FORALL or GET statement must be qualified with the clause WHERE @HANDLE@ = <handle> & <actual-where>. In this clause, <handle> stands for the value of the second parameter passed to the event rule and <actual-where> stands for a WHERE clause, possibly empty (if it is empty no ampersand (&) should be specified). Alternatively, the data parameter notation can be used. Refer to Sample Event Rule on page 94.

Passed Parameters

The following parameters are passed to the stored procedure event rules:

- The name of the SLK table that maps the stored procedure currently executing (IC16)

- An internal handle (B4)
- A result count (B4)

  For row-count-returned rules, if the result count (parameter 3) is zero, the number of rows affected is indeterminate; otherwise, it represents the number of rows affected.

  For cursor-returned event rules, the result count (parameter 3) represents the number of columns in the result set.

- An event count (B4)

  The event count (parameter 4) denotes the actual sequential number of the event being processed within its group (row-count-returned or cursor-returned).

Sample Invocation

Sample Stored Procedure

The following is a stored procedure as defined in MS SQL Server:

```
create procedure sstub (@p1 decimal(3)=9,@p2 decimal (3)=6 output) as
declare @pp decimal (3)
select @pp=@p2
select title_id,pubdate,royalty from pubs.dbo.titles where royalty <36
select title_id,royalty from pubs.dbo.titles where royalty >36
select @p2=@p2-@p1
return @pp
```

Sample Stored Procedure Mapping

The following is a stored procedure mapping in a SLK definition:

```
COMMAND==>                              TABLE DEFINITION

      Table: SQLL             Type: SLK     Unit: HURON

   DBSource: sqls                                       ServerType: SLK
   ServerId: SLK2                                    Proc/ProcResSet: P (P/R/N)
  Procedure: sstub;1                                    ServerOrders: Y (Y/N)
  Qualifier: pubs                                     ImpliedUpdates: N (Y/N)
      Owner: dbo                                          Dictionary: X (X/I)

  Location Parm    Default        Src Sourcename         Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                              _
   |--- External Field --------------|--- Metadata Field -------------------
    ExternalName                  Vrt Name           Typ Syn Len  Dec Ord Req
    ------------------------------ -  ---------------- - -- -----  --  -   -
K _ @HANDLE@                         @HANDLE@         Q B      2   0
K _ @RESULT@                         @RESULT@         Q B      2   0
S _ RETURN_VALUE                   R  RETURN_VALUE     Q P      6   0
S _ @p1                            I  @P1              Q B      2   0
S _ @p2                            B  @P2              Q B      2   0
(P - parameter K - key, S - select,  I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=PROCS 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

Sample Result Set Mappings

The stored procedure STUB builds two result sets. We map the first of them with the following SLK definition:

```
COMMAND==>                              TABLE DEFINITION

      Table: SQLL1             Type: SLK     Unit: HURON

   DBSource: sqls                                    ServerType: SLK
   ServerId: SLK2                                 Proc/ProcResSet: R (P/R/N)
  Procedure: titles                                  ServerOrders: Y (Y/N)
  Qualifier: pubs                                   ImpliedUpdates: N (Y/N)
      Owner:     dbo                                    Dictionary: X (X/I)

  Location Parm    Default        Src Sourcename          Event Rule     Typ Acc
  ---------------- ---------------- - ----------------   ---------------- -  -
_ LOCATION                                                _
   |--- External Field --------------|--- Metadata Field --------- ----------
    ExternalName                   Vrt Name            Typ Syn Len  Dec Ord Req
    ------------------------------ -  ---------------- - -- -----  --  -   -
P _ @HANDLE@                          @HANDLE@          Q B      2   0
S _ title_id                        S  TITLE_ID         S V      6   0
S _ pubdate                         S  PUBDATE          D B      4   0
S _ royalty                         S  ROYALTY          Q P      6   0
  _
(P - parameter, K - key, S - select,  I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=PROCS 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

We map the second result set with the following SLK definition:

```
COMMAND==>                              TABLE DEFINITION

      Table: SQLL2              Type: SLK     Unit: HURON

   DBSource:  sqls                                        ServerType: SLK
   ServerId:  SLK2                                  Proc/ProcResSet: R (P/R/N)
  Procedure: titles                                    ServerOrders: Y (Y/N)
  Qualifier: pubs                                    ImpliedUpdates: N (Y/N)
      Owner:     dbo                                        Dictionary: X (X/I)

Location Parm    Default          Src Sourcename       '  Event Rule      Typ Acc
  -------------- ----------------  -  ---------------- '  ---------------- -   -
_ LOCATION
   |--- External Field ------------------|-------------- Metadata Field ---------
   ExternalName                        Vrt Name        Typ Syn Len  Dec Ord Req
    ------------------------------ -  ---------------- -  -- -----  --  -   -
P _ @HANDLE@                            @HANDLE@     Q  B      2   0
S _ title_id                             S  TITLE_ID     S  V      6   0
S _ royalty                              S  ROYALTY      Q  B      4   0
_
_
(P - parameter, K - key, S - select,  I - insert, R - replicate, D - delete)
PFKEYS: 3=SAVE 22=DEL 13=PRINT 2=DOC 4=PROCS 5=COLUMNS 6=QUALIFIERS 9=OWNERS
```

For stored procedure result set mappings, external table characteristics (name, qualifier, owner) are ignored at runtime; however, it is helpful sometimes to have these values in the definition. In the example, they assist in defining the mapping, as they refer to the table used by the procedure to build the result set.

Sample Event Rule

The following event rule is executed when a result set is returned (Access=U):

```
RULE EDITOR ==>
_  C(PROC, HANDLE, RESULT, COUNT);
_ ------------------------------------------------------------------------------
_ COUNT = 1;                                                      | Y N N
_ COUNT = 2;                                                      |   Y N
  ----------------------------------------------------------------+---------
_ FORALL SQLL1 WHERE @HANDLE@=HANDLE :                            | 1
_    CALL MSGLOG(SQLL1.TITLE_ID ||','|| SQLL1.PUBDATE) || ','     |
_       || SQLL1.ROYALTY);                                        |
_    END;                                                         |
_ FORALL SQLL2(HANDLE) :                                          |   1
_    CALL MSGLOG(SQLL2.TITLE_ID ||',' || SQLL2.ROYALTY);          |
_ END;                                                            |
  ------------------------------------------------------------------------------
```

Sample Invocation Statements

Syntax	Alternative Syntax
CALL STUB(2,3)	CALL STUB WHERE @P1=2 & @P2=3
CALL STUB(2,'')	CALL STUB WHERE @P1=2
CALL STUB('',3)	CALL STUB WHERE @P2=3
CALL MSGLOG(STUB(2,3)) [a]	
EXECUTE STUB(2,3)	EXECUTE STUB WHERE @P1=2 & @P2=3
EXECUTE STUB(2,'')	EXECUTE STUB WHERE @P1=2
EXECUTE STUB('',3)	EXECUTE STUB WHERE @P2=3
TRANSFERCALL STUB(2,3)	TRANSFERCALL STUB WHERE @P1=2 & @P2=3
TRANSFERCALL STUB(2,'')	TRANSFERCALL STUB WHERE @P1=2
TRANSFERCALL STUB('',3)	TRANSFERCALL STUB WHERE @P2=3

a. This is possible because the procedure returns a value.

After EXECUTE and TRANSFERCALL statements, you can use the statement CALL MSGLOG(GETENMSG) to extract the return value produced by the procedure. GETENDMSG returns a NULL if the definition contains no return value. In all cases, upon return of control after the execution of the procedure, the current occurrence of the SLK table STUB contains the values of the procedure parameters (including the return value) as set by the procedure.

Appendix A Documenting TIBCO Object Service Broker SLK Tables

This appendix provides information to document the TIBCO Object Service Broker SLK table.

Topics

Using the Documentation Screen

Each table definition in TIBCO Object Service Broker has a Documentation screen associated with it. You use this screen to create or modify documentation for the table. To display the Documentation screen for a TIBCO Object Service Broker SLK table, press PF2 from the Table Definer.

Sample Documentation Screen

```
DESCRIPTION OF TABLE       SLKDEMO                UNIT: DOC

MODIFIED ON 03 MAR 00  BY  USR40     CREATED ON 29 SEP 99 BY USR40

KEYWORDS DEMO EXTERNAL DATA SLK
SUMMARY Demo SLK table accessing external Oracle data

                         Description
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_ This table contains demo information for use with SLK tables.
_




PFKEYS: 3=END  5=VIEW DOCUMENT  13=PRINT  12=EXIT
```

Field Values

The Table Definer updates some of the fields on this screen. You must maintain the **KEYWORDS**, **SUMMARY**, and **DESCRIPTION** fields.

KEYWORDS	Type individual words that briefly describe the table. These words are used by the Keyword Search facility in TIBCO Object Service Broker. This field is one line long and can contain multiple entries, separated by commas or blanks.

SUMMARY	Type a one line summary of the **DESCRIPTION** field.
DESCRIPTION	Type information about the table (for example, what its role is, what it does, and how it works) using TIBCO Object Service Broker **`SCRIPT`** commands. There is no limit to the amount of information you can type in this field.

PF Keys

The following PF keys are supported from the Documentation screen:

PF1	Displays corresponding help for the field or screen where your cursor is placed.
PF3	Saves changes and returns you to the Table Definer.
PF5	Toggles between browse and edit modes.
PF12	Cancels changes and returns you to the Table Definer.
PF13	Prints the version of the documentation that you are viewing.

See Also *TIBCO Object Service Broker Shareable Tools* for more information on the SCRIPT tool.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W