TIBCO Service Gateway™ for DB2

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 DB2 data and define TIBCO Object Service Broker tables based on this data.

This manual describes the TIBCO Object Service Broker interface to DB2 data; the interface is known as the TIBCO Service Gateway for DB2.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Object Service Broker Documentation

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

Fundamental Information

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

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

Application Development and Management

The following manuals provide information about application development and management:

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

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

System Administration on the z/OS Platform

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

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

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

System Administration on Open Systems

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

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

External Database Gateways

The following manuals describe external database gateways:

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

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

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *OSB_HOME*	By default, all TIBCO products are installed into a folder referenced in the documentation as *TIBCO_HOME*. On open systems, TIBCO Object Service Broker installs by default into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *OSB_HOME*. The default value of *OSB_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\OSB`. Similarly, all TIBCO Service Gateways on open systems install by default into a directory in *TIBCO_HOME*. For example on Windows systems, the default value is `C:\tibco\OSBgateways\6.0`. On z/OS, no default installation directories exist.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

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

Table 2 Syntax Typographical Conventions

Convention	Use

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

Connecting with TIBCO Resources

How to Join TIBCOmmunity

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

How to Access All TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

Chapter 1 Installing Service Gateway for DB2

This chapter introduces Service Gateway for DB2, and provides the information to install, configure, and customize the software.

Topics

Introduction

Service Gateway for DB2 is a server interface to DB2 data that you use for concurrent real-time access to DB2 data from TIBCO Object Service Broker. It can execute only in a z/OS environment and connect to z/OS DB2.

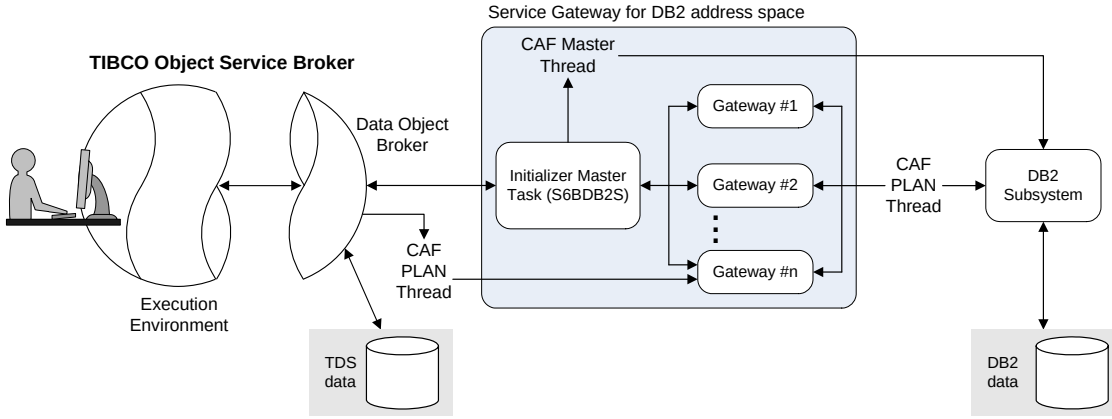
The Service Gateway for DB2 interface consists of the following components:

- **Table Definer** – Define DB2 tables.
- **External server** – Access DB2 table definitions using TIBCO Object Service Broker DB2 tables, or access DB2 data when TIBCO Object Service Broker data access is requested for a DB2 table.

Initially, the Gateway generates dynamic Structured Query Language (SQL) to access DB2 data. When your TIBCO Object Service Broker application is complete, you can collect TIBCO Object Service Broker DB2 access statements and use them to generate static SQL, as described in Collecting TIBCO Object Service Broker DB2 Access Statements on page 121. Generating static SQL is described in Chapter 3, Using Static SQL, on page 61.

Accessing DB2 Data

The figure below shows how you can access DB2 data, while still having access to TDS data, which is TIBCO Object Service Broker's native data type.



The Gateway ensures that data is presented to TIBCO Object Service Broker rules in a manner consistent with TIBCO Object Service Broker behavior. You can access DB2 data by doing the following:

- Define a TIBCO Object Service Broker table of type DB2 based on a DB2 table definition.
- Access DB2 data from TIBCO Object Service Broker.

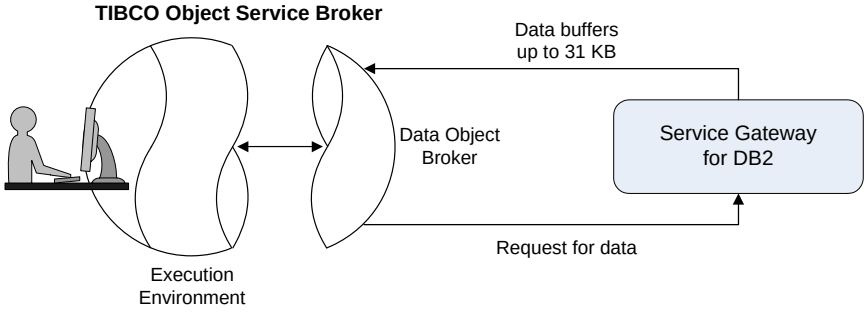
Initializing a Gateway

The initializer program (S6BDB2M) is passed a number of parameters, including one that instructs it to attach a specified number of Gateway tasks, allowing multiple Gateways in a single Gateway address space. The initializer program is used only to start up a Gateway and maintain the Gateway subtasks. It establishes communication to the Data Object Broker using the TIBCO Object Service Broker Communication Subsystem and to the DB2 subsystem using the Call Attach Facility (CAF) CONNECT.

Each Gateway task establishes a task-level connection to TIBCO Object Service Broker. When the Execution Environment requests access to DB2 data, a CAF OPEN PLAN is issued. Gateway parameters are used to determine the length of time that the connection to the DB2 subsystem is active (PLAN stays open). The Gateway supports read/write access to DB2 data.

Processing DB2 Data

The following illustration shows how 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.



Deployment

You can configure the Data Object Broker and the Service Gateway for DB2 to reside on different hosts and/or operating systems (z/OS, Windows, or Solaris). Service Gateway for DB2 must be in the same logical partition as the DB2 database system. Refer to Connecting the Gateway to a Windows, or Solaris Data Object Broker on page 58 for additional information.

If all components reside in the logical partition and in authorized libraries, Cross Memory Services is used for communications.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating or *TIBCO Object Service Broker for Open Systems Installing and Operating* for more information on communications requirements.

Preparing for Installation

If you are installing this product on a remote host in relation to the Data Object Broker, where access to the product is only via a network, see Installing on a Remote Host on page 11.

Before installing Service Gateway for DB2, review the following:

- **TIBCO Object Service Broker Base Component** – You must install and ACCEPT (using SMP/E) the TIBCO Object Service Broker base component before installing Service Gateway for DB2. You must also have the `<HLQ>.INSTALL` data set that was created during that installation.
- **Supported Versions of DB2** – Refer to the Late Breaking News link on our http://support.tibco.com/ web site for the most current information about the levels, versions, and releases of DB2 that Service Gateway for DB2 supports.
- **Customizing the OSEMOD DB2 Variables** – Member OSEMOD in the CLIST data set is an ISPF edit macro used to customize members in the DB2.JOBS, CLIST, CNTL and JCL data sets. Refer to *TIBCO Object Service Broker for z/OS Installing and Operating* for the DB2 installation variables in OSEMOD that you should customize as required.

Distribution Media and Contents

This section describes how to obtain the software, and the installation file that comprises the distribution media. Similar to the TIBCO Object Service Broker base component, the Service Gateway for DB2 software is distributed in `.xml` format within a zip file.

Distribution File Format

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

Obtaining the Installation Media

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

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

Installation Files

The following zip file comprises the distribution media:

```
TIB_srvcgw-DB2_6.0.0_zos.zip
```

Uploading the Software

If you have acquired Service Gateway for DB2 by downloading it from the TIBCO Software web site, you must upload the software to the z/OS host system.

Preparing and Uploading the Product File

1. Download or copy the `TIB_srvcgw-DB2_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.
2. Unzip the file to a temporary location on the PC. The zip file contains multiple files; of these, the following file is the only file used in this installation:

   `db2.xm1` – compressed file containing Service Gateway for DB2

The `srvcgw_db2.xm1`, `install.bin`, `ostarrec.bin`, `property.bin`, and `OSTAREDC` files are not used in this procedure.

3. Pre-allocate the following sequential data set on the z/OS host system:

   ```
   <HLQ>.DB2.XM1                          (size 18 KB)
   ```

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

   ```
   //ALLOC EXEC PGM=IEFBR14
   //DD1   DD DSN=<HLQ>.DB2.XM1,
   //       DISP=(,CATLG,DELETE),UNIT=SYSDA,
   //       DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
   //       SPACE=(TRK,(2,1))
   ```

4. FTP the `db2.xm1` file in BIN mode to the `<HLQ>.DB2.XM1` data set.

Installing the Software

You must perform the installation under an ISPF environment only.

This section describes the procedure for installing the Service Gateway for DB2.

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

- <HLQ>.INSTALL
- <HLQ>.DB2.XM1

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

Installation Overview

To install Service Gateway for DB2, perform the following:

1. Edit the properties file by specifying the keywords for installing this component.
2. Install the software.

Edit the Properties File

Edit the PROPERTY member in <HLQ>.INSTALL. Table 3 describes keywords in the properties file for installing this component.

Table 3 Properties File Keywords

Keyword	Description
INSTALL=	To install Service Gateway for DB2, specify DB2: INSTALL=DB2
SYSDB2=	Fully qualified DB2 load library.
DB2PLN=	Name for a DB2 plan used in Service Gateway for DB2.

Initial Installation

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

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

The DB2.JCL data set is created at the successful completion of this step.

STEP 2: **Run Job DB2.JCL.**

This batch job will uncompress the DB2.XM1 file to produce the distribution library.

JCL in: `<HLQ>.DB2.JCL (Edit JOB card to your site's standards)`

Data Set: `<HLQ>.DB2.JCL (SUB data set)`

Uncompressing `<HLQ>.DB2.XM1` produces the distribution library `<HLQ>.DB2.FILEI`.

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

Member in: `<HLQ>.DB2.FILEI`

Member: `S6C1CUST  (EX member)`

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

`Customized copy – Library Description`

- `<HLQNONV>.<INSTVER>.JCL – Sample JCL`
- `<HLQNONV>.<INSTVER>.DB2.JOBS – Install jobs for DB2`

STEP 4: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.DB2.JOBS`

Member: `S6C2RUNJ (EX member)`

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

This completes the installation process for DB2. See Starting the Gateway on page 32 for details on starting a DB2 Gateway.

You can modify the STATUS of any job as per your requirement. For example, if your shop normally ACCEPTs the product FMID at some future time, then change the status of S6C4ACPT from INSTALL to FUTURE. Note that you must ACCEPT the Service Gateway for DB2 component before applying any hotfix maintenance using SMP/E.

Installing on a Remote Host

This section describes the procedure for installing the software on a remote host in relation to the Data Object Broker installation.

Distribution Media and Contents

This software is distributed in `.xml` format within a zip file. The file is in a format compatible with IBM System Modification Program/ Extended (SMP/E) naming conventions. The software is packaged in SMP/E txlib format.

Obtaining the Installation Media

You can download the software from the TIBCO web site by following these steps:

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

Installation Files

The following zip file comprises the distribution media:

`TIB_srvcgw-DB2_6.0.0_zos.zip`

Uploading the Software

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

Preparing and Uploading the Product File

1. Download or copy the `TIB_srvcgw-DB2_6.0.0_zos.zip` file to a PC that can connect to the z/OS host system.

2. Unzip the file to a temporary location on the PC. The zip file contains multiple files; of these, the following files are the only files used in this installation:
   - `srvcgw_db2.xm1` – compressed file containing Service Gateway for DB2 for installation on a remote host
   - `install.bin` – the REXX EXEC to perform the installation
   - `ostarrec.bin` – the REXX EXEC to uncompress the `.xm1` file
   - `property.bin` – a template of mandatory install variables required for product installation.
   - `OSTAREDC` – a load module to improve the performance of `OSTARREC`

The `db2.xm1` file is not used in this procedure.

3. Pre-allocate a PDS, fixed block data set on the z/OS host system with the following name:

   ```
   <HLQ>.INSTALL
   ```

   where <HLQ> is any valid high-level qualifier. Note that this <HLQ> will be used during the installation. See the sample JCL in the next step.

4. Pre-allocate the following sequential data set on the z/OS host system:

   ```
   <HLQ>.OS.DB2.XM1                    (size 46,220 KB)
   ```

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

   ```
   //ALLOC EXEC PGM=IEFBR14
   //DD1   DD DSN=<HLQ>.INSTALL,
   //       DISP=(,CATLG,DELETE),UNIT=SYSDA,
   //       DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
   //       SPACE=(TRK,(5,15,100))
   //DD2   DD DSN=<HLQ>.OS.DB2.XM1,
   //       DISP=(,CATLG,DELETE),UNIT=SYSDA,
   //       DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0,DSORG=PS),
   //       SPACE=(TRK,(1000,50))
   ```

5. FTP `install.bin`, `property.bin` and `ostarrec.bin` to your z/OS system in BIN mode to the `<HLQ>.INSTALL` data set. Name these utilities `INSTALL`, `PROPERTY` and `OSTARREC`, respectively.

6. FTP the `srvcgw_db2.xm1` file in BIN mode to the `<HLQ>.OS.DB2.XM1` data set.

Installing the OSTAREDC Program

1. Upload the OSTAREDC file to z/OS in binary format to a data set with LRECL=80 and RECFM=FB.
2. In ISPF 3.4, against this data set, type the following:

   ```
   "RECEIVE INDA(/)"
   ```

   When prompted, specify DA('<HLQ>.INSTLOAD' as the name of the load library where you want the OSTAREDC program restored, using the following syntax:

   ```
   DA('datasetname')
   ```

3. Edit OSTARREC as follows:
   - Issue the command "FIND OSTAREDC 1".
   - Change the constant after the equal sign to contain the full data set name of the program. The string must start with a double quote and a single quote, and end with a single quote and a double quote (the double quotes delimit the string and the single quotes tell TSO that the data set name is fully qualified). For example, change the following:

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

     to

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

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

Installing the Software

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

- <HLQ>.INSTALL
- <HLQ>.OS.DB2.XM1

The <HLQ> referenced throughout this chapter is the high-level qualifier you specified when you uploaded the product software. This is the value of the INSTALL and XM1 files you specified. It will be used as the default value for all distribution files created when an XM1 is uncompressed. It is equivalent to the value of symbolic parameter HLQ as described in OSEMOD.

Installation Overview

To install Service Gateway for DB2, perform the following:

1. Determine your system environment values listed in System Environment Checklist.
2. Edit the Properties File using the values determined in Step 1.
3. Install the Software.

System Environment Checklist

Before you begin the installation, review the system environment information described in Table 4 and determine whether you will use the default value or provide your own value.

Table 4 OSEMOD Variables

Description	OSEMOD Variable	Default Value	Your Value
High level qualifier for uploaded data sets INSTALL and OS.DB2.XM1.	HLQ	Specified on upload	
High level qualifier for non-VSAM and VSAM data sets you are authorized to create.	$HLQNONV$ $HLQVSAM$	TIBCO.TESTNV TIBCO.TESTVS	
Second level qualifier for install files.	$INSTVER$	INS60	
Second level qualifier for TIBCO Service Gateway system files.	SLQ	OSB60	
Second level qualifier for SMP/E libraries	SMP	SMP60	
For SMS Shops – managementclass, dataclass and storageclass, if required			
For new non-VSAM data sets.	$NMGTCLAS $NDATCLAS$ $NSTOCLAS$	STANDARD STANDARD S6BNONV	
For new VSAM data sets.	$VMGTCLAS $VDATCLAS$ $VSTOCLAS$	STANDARD STANDARD S6BVSAM	

Table 4 OSEMOD Variables

Description	OSEMOD Variable	Default Value	Your Value
High level qualifier of Language Environment libraries for SCEELKED and SCEEBIND.	$CEELIB$	CEE	
High level qualifier of IBM's Callable Services library CSSLIB.	$CSSLIB$	SYS1	
Fully qualified DB2 load library.	$SYSDB2$	DSN810.SDSNLOAD	
Name for a DB2 plan	$DB2PLN$	OSBPLAN	

For additional information, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Edit the Properties File

Use the PROPERTY member in <HLQ>.INSTALL as a template, and modify to suit your requirements. Table 5 describes keywords in the properties file that correspond to the system environment variables in System Environment Checklist.

Table 5 Properties File Keywords

Keyword	Description
`INSTALL=`	To install Service Gateway for DB2, specify `REMOTEGATEWAY`: `INSTALL=REMOTEGATEWAY`
`SERVICEGATEWAY=`	To install Service Gateway for DB2, specify `DB2`: `SERVICEGATEWAY=DB2`
`HLQNONV=`	High level qualifier for non-VSAM data sets.
`HLQVSAM=`	High level qualifier for VSAM data sets.
`INSTVER=`	Second level qualifier for install files.
`SLQ=`	Second level qualifier for TIBCO Object Service Broker system files.
`SMP=`	Second level qualifier for SMP/E libraries.

Table 5 Properties File Keywords

Keyword	Description
SMS=	YES for SMS site, NO for non-SMS site. **Warning:** If you select the SMS=YES option, be sure to specify SMS-managed data-set names. The SMS automatic class selection (ACS) rules for your site determine whether a data-set name is eligible for SMS management. The name you specified that is determined by the ACS routines to be SMS-eligible is SMS-managed. Otherwise, the result is unpredictable.
COMPAT=	Use if SMS=YES. Valid values: YES for SMS compatible data set name classes; NO for SMS non-compatible data set name classes. If COMPAT=NO, specify the following: • NMGTCLAS – MANAGEMENTCLASS for non-VSAM data sets • NDATCLAS – DATACLASS for non-VSAM data sets • NSTOCLAS – STORAGECLASS for non-VSAM data sets • VMGTCLAS – MANAGEMENTCLASS for VSAM data sets • VDATCLAS – DATACLASS for VSAM data sets • VSTOCLAS – STORAGECLASS for VSAM data sets
VOLSER=	If SMS=YES, specify one DASD volume for VSAM data set allocation. Default is USER01. If SMS=NO, specify three DASD volumes separated by commas. Defaults are OSBS06, OSBD18, OSBB02. • vol1 – DASD volser for temp work files • vol2 – DASD volser for install files • vol3 – DASD volser for TIBCO Object Service Broker system files
CEELIB=	High level qualifier of Language Environment libraries.
CSSLIB=	High level qualifier of IBM's Callable Services library CSSLIB.
SYSDB2=	Fully qualified DB2 load library.
DB2PLN=	Name for a DB2 plan used in Service Gateway for DB2.

Install the Software

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

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

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

Member in: `<HLQ>.INSTALL`

Member: `INSTALL (EX member)`

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

The DB2.JCL data set is created at the successful completion of this step.

STEP 2: **Edit the Job card to your site's standards and run Job DB2.JCL.**

JCL in: `<HLQ>.DB2.JCL (Edit Job Card to your site's standards)`

Data Set: `<HLQ>.DB2.JCL (SUB data set)`

This batch job will uncompress the OS.DB2.XM1 file to produce the distribution libraries. If you modify the job name, make sure it does not exceed seven characters. The job should successfully complete with a return code of 0.

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

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

Member in: `<HLQ>.FILECLS`

Member: `OSEMOD`

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

Member in: `<HLQ>.OS.DB2.FILEI`

Member: `S6P1CUST (EX member)`

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

`Customized copy – Library Description`

- `<HLQNONV>.<INSTVER>.CLIST – CLIST`
- `<HLQNONV>.<INSTVER>.CNTL – CNTL`
- `<HLQNONV>.<INSTVER>.JCL – Sample JCL`
- `<HLQNONV>.<INSTVER>.OS.DB2.JOBS – Install jobs for remote Service Gateway for DB2`

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

Member in: `<HLQNONV>.<INSTVER>.OS.DB2.JOBS`

Member: `JOBSP (EDIT member)`

Jobs in MEMBER are evaluated in the order they are listed and are submitted based upon their specified STATUS. The next job is submitted only if the previous one completed with its expected return code RC.

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

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

You can modify the STATUS of any job as per your requirement. For example, if your shop normally ACCEPTs the product FMID at some future time, then change the status of S6P4ACPT from INSTALL to FUTURE. Note that you must ACCEPT the remote Service Gateway for DB2 FMID before applying any hotfix maintenance using SMP/E.

Skip this step if the default STATUS of all the jobs is acceptable.

STEP 6: **Initiate install jobs.**

Member in: `<HLQNONV>.<INSTVER>.OS.DB2.JOBS`

Member: `S6P2RUNJ (EX member)`

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

This completes the installation process. See Starting the Gateway on page 32 for details on starting a DB2 Gateway.

STEP 7: **Connect the Service Gateway for DB2 to the DOB.**

Modify the relay configuration file for the DB2 server on z/OS, and huron.dir on the DOB, as documented in Connecting the Gateway to a Windows, or Solaris Data Object Broker, page 58.

Uninstalling on a Remote Host

To uninstall the software, perform the following:

STEP 1: **Run the DB2 cleanup job.**

Member in: `<HLQNONV>.<INSTVER>.OS.DB2.JOBS`

Member: `S6P9CLEN (Edit JOBCARD and SUB member)`

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

1. `$HLQ$.FILECLS`
2. `$HLQ$.FILECTL`
3. `$HLQ$.FILEEM1`
4. `$HLQ$.FILEEM2`
5. `$HLQ$.FILEJCL`
6. `$HLQ$.FILEOBJ`
7. `$HLQ$.FILETRK`
8. `$HLQ$.FILEXML`
9. `$HLQ$.MACRO`
10. `$HLQ$.DBRM`
11. `$HLQ$.OS.DB2.FILEI`
12. `$HLQNONV$.$INSTVER$.CLIST`
13. `$HLQNONV$.$INSTVER$.CNTL`
14. `$HLQNONV$.$INSTVER$.JCL`
15. `$HLQNONV$.$INSTVER$.OS.DB2.JOBS`

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

- `$HLQ$.INSTALL`
- `$HLQ$.OS.DB2.XM1`

Manually delete the data sets.

Binding the Gateway Plan

After linking, the Gateway plan must be bound.

Determining Binding Procedure

How the Gateway plan is bound depends on whether you are using static SQL. To determine if you are using static SQL, view the tables of type DB2 and any static SQL information using the @STATICSQL tool. For more information about @STATICSQL, refer to Chapter 3, Using Static SQL, on page 61.

Binding With Static SQL or Stored Procedures

Member BINDDB2S in the JCL data set (<HLQNONV>.<INSTVER>.JCL) binds the Gateway plan using members XBINDDB5 and XBINDDB7 of the CNTL data set (<HLQNONV>.<INSTVER>.CNTL) as input. The DB2 subsystem must be active to bind the Gateway plan.

BINDDB2S also references member XBINDDB6 in the data set defined in the @DB2SERVERJCL export table. XBINDDB6 (automatically generated by @STATICSQL each time you generate or remove static SQL) identifies additional DBRMs to include in your plan.

To bind the plan, do as follows:

1. If you are binding a Gateway for the first time, customize members BINDDB2S and XBINDDB5.

   Member XBINDDB5 identifies the DB2 subsystem and the Gateway plan name, as shown in the sample below:

```
DSN SYSTEM($DB2SSI$)
BIND PLAN($PLNAME$) -
  MEMBER(HDB2SREQ,HDB2SEL,HDB2ISRT,HDB2REPL,HDB2DLET,HDB2STPR -
    STATIC SQL HANDLERS if any
    STORED PROCEDURES if any
    .
    .
    .
```

   The value of $PLNAME$ *must* be the same as the value of the PLAN gateway parameter. Modify $PLNAME$ as required.

   If you are using the BROWSEPLAN and UPDATEPLAN gateway parameters, you must run this job once more for each of these parameters that you use, changing the PLAN gateway parameter accordingly. For more information,

refer to Specifying the Gateway Plan to TIBCO Object Service Broker on page 26.

2. Review member XBINDDB7 in the CNTL data set and customize if required. XBINDDB7 contains the remainder of the bind parameters as shown in the following sample:

```
) -
ACTION(REPLACE) RETAIN -
VALIDATE (RUN) -
ISOLATION (RR) -
ACQUIRE(USE) -
RELEASE(COMMIT)
```

3. Submit BINDDB2S.

   It should end with RC=0.

Sample bind parameters use ISOLATION(RR) for integrity, ACQUIRE(USE) and RELEASE(COMMIT) for maximum concurrency, and VALIDATE(RUN) for static SQL purposes. This is consistent with TIBCO Object Service Broker processing.

ISOLATION(CS) could lose data integrity during updates, because when DB2 releases locks on data, the data could still reside in a TIBCO Object Service Broker buffer. You can view the original data; be aware that data whose locks were released by a DB2 application or by a TIBCO Object Service Broker transaction could have changed. If you change the ISOLATION parameter to Cursor Stability (CS), ensure that TIBCO Object Service Broker pseudo-conversational coding techniques are used by applications updating DB2 data or that prior to an update the data row is required if you are unsure if the cursor is still positioned on the current DB2 page.

Consider using the DB2 BIND PACKAGE subcommand to manage the binding of Static SQL handlers. If you use it, you can have different ISOLATION and RELEASE parameters for each handler and it alleviates rebinding the entire Gateway plan each time a new Static SQL handler is introduced.

Binding Without Static SQL

Member BINDDB2 in the JCL data set binds the Gateway without static SQL using members XBINDDB5 and XBINDDB7 of the CNTL data set as input. The DB2 subsystem must be active to bind the Gateway plan.

To bind the Gateway without Static SQL, complete the following steps:

1. If binding a Gateway for the first time, customize members BINDDB2 and XBINDDB5.

Member XBINDDB5 identifies the DB2 subsystem and the Gateway plan name as shown in the sample in Binding With Static SQL or Stored Procedures on page 21.

The PLAN parameter *must* correspond with the name provided by the PLAN gateway parameter. Modify this name as necessary.

If you are using the BROWSEPLAN and UPDATEPLAN gateway parameters, you must run this job once more for each of these that you use, changing the PLAN parameter accordingly.

2. Review member XBINDDB7 in the CNTL data set and customize if required.

   XBINDDB7 contains the remainder of the bind parameters as shown in the sample member in Binding With Static SQL or Stored Procedures on page 21.

   Sample bind parameters use ISOLATION(RR) for integrity, ACQUIRE(USE) and RELEASE(COMMIT) for maximum concurrency, and VALIDATE(RUN) for static SQL purposes (modify this parameter to VALIDATE(BIND). This is consistent with TIBCO Object Service Broker processing.

   ISOLATION(CS) could lose data integrity during updates, since when DB2 releases locks on data, the data could still reside in a TIBCO Object Service Broker buffer. You can view the original data; be aware that data whose locks were released by a DB2 application program or by a TIBCO Object Service Broker transaction could have changed. If you change the ISOLATION parameter to Cursor Stability(CS), ensure that TIBCO Object Service Broker pseudo-conversational coding techniques are used by applications updating DB2 data.

3. Submit BINDDB2.

   It should end with RC=0.

Requirements for Accessing DB2 Using Static SQL

Required Static SQL Data Sets

The following data sets must be allocated on every z/OS image where Service Gateway for DB2 will generate or use static SQL to access the DB2. The ALLOCDB2 member in the JCL data set distributed with TIBCO Object Service Broker provides sample JCL to allocate these data sets.

Allocated by...	Allocated in the Images Occupied by ...	Description
DD1	TIBCO Object Service Broker and DB2	An assembler source data set for the static SQL handlers generated by the @STATICSQL tool, specified in the **File** field for the @DB2SERVERASM export table definition.
DD2	DB2	An object code data set for assembled static SQL handlers.
DD3	DB2	A DBRM library to hold DBRMs generated when the static SQL handlers are assembled.
DD4	DB2	A load library for static SQL handlers (does not have to be authorized); use the //HDB2SSQL DD statement to point to this library.
DD5	TIBCO Object Service Broker and DB2	A data set for JCL generated by the @STATICSQL tool, which assists in assembling and linking the SQL handlers, and binding the Gateway. This data set is specified in the **File** field for the @DB2SERVERJCL export table definition.
DD6	TIBCO Object Service Broker and DB2	A source listing data set for the static SQL handler assembler listings.

Using the @STATICSQL Tool

Before you run @STATICSQL for the first time, the @DB2SERVERJCLMDL, @DB2SERVERJCL, and @DB2SERVERASM import and export tables must be modified.

To generate static SQL, collect your TIBCO Object Service Broker DB2 access statements by running your applications in an Execution Environment with the DB2LOG Execution Environment parameter set to Y. The @STATICSQL tool can then be run to generate the static SQL handlers and the JCL to assemble and link the handlers.

Refer to Chapter 3, Using Static SQL, on page 61 for more information.

See Also *TIBCO Object Service Broker Parameters* for further information on parameters.

Specifying the Gateway Plan to TIBCO Object Service Broker

Generally you specify the DB2 Gateway plan using the PLAN gateway parameter.

Specifying Different DB2 Plans

You can specify a browse plan name and an update plan name for a group of Gateways. To do this, you use the following parameters:

```
BROWSEPLAN=browseplanname
UPDATEPLAN=updateplanname
```

If the Gateway wants to use the value of either of these gateway parameters and that value is not specified, the Gateway uses the name specified for the PLAN gateway parameter.

The Gateway uses the browse flag passed in the first message of a transaction to determine which plan to use. If the browse flag is on, the Gateway uses the browse plan, otherwise it uses the update plan. For example, if you browse a DB2 table from the table browser, the message sent is set to browse mode and the Gateway uses the browse plan. If you then select a row to invoke the single occurrence editor, another Gateway is assigned, that transaction does not have the browse flag set, and that Gateway uses the update plan.

Scope Considerations

Specifying a SCOPE of other than TRANSACTION when specifying separate browse and update plans generates an error in the Gateway start up. Only a SCOPE of TRANSACTION closes the plan at the end of a transaction. A scope of SESSION or time does not close the plan at transaction end and the next transaction inherits the previous transaction's plan.

SECLEVEL=1 forces a SCOPE of TRANSACTION.

SECLEVEL=0 defaults to a scope of SESSION. It can be overridden by using the SCOPE=TRANSACTION parameter.

Isolation Considerations

A browse plan should specify an isolation level of Cursor Stability.

An update plan should specify an isolation level of Repeatable Read.

A plan specifying an isolation level of Cursor Stability can still perform updates.

Implementing Security

The Gateway provides two methods of authorizing access to DB2 data:

- DB2 Subsystem
- External security interface

The method used is determined by the SECLEVEL parameter specified at startup time. For details, see Gateway Parameters on page 42.

Methods of Authorizing Access to DB2 Data

DB2 Subsystem (SECLEVEL=0)	The DB2 subsystem verifies all DB2 table accesses using the ID that started the Gateway (if the Gateway is running as a batch job) or the started task name (if the Gateway is running as a started task). This ID or started task name requires the security described in DB2 Security below.
External security interface (SECLEVEL=1)	DB2 table access is verified using the TIBCO Object Service Broker session ID or the current group name. The external security interface is implemented using the sample IDENTIFY AUTHORIZATION EXIT DSN3@ATH. Refer to Implementing External Security on page 52 for more information. TIBCO Object Service Broker session IDs or group names require security as described in DB2 Security below.

DB2 Security

The following table lists required DB2 security for the listed tasks:

Task	DB2 Security
Defining TIBCO Object Service Broker DB2 tables	EXECUTE authorization on the Gateway plan. SELECT authorization on SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, @SYSROUTINES and @SYSPARMS.
Accessing DB2 data	EXECUTE authorization on the Gateway plan. Appropriate authorization (SELECT, INSERT, UPDATE, DELETE or ALL) to all DB2 tables to be accessed from TIBCO Object Service Broker using Dynamic SQL. **Note** After generating Static SQL for a DB2 table, the Gateway always attempts to access this table using Static SQL. Dynamic SQL is used as described in Conditions Under Which Dynamic SQL is Used on page 76. When Static SQL is used, only EXECUTE authority on the Gateway plan is required. When Dynamic SQL is used, both EXECUTE authority on the Gateway plan and individual table authorizations are required.

TIBCO Object Service Broker Security

To restrict the ability to define TIBCO Object Service Broker DB2 tables, restrict read access to the TIBCO Object Service Broker tables that map to SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, SYSIBM.SYSROUTINES and SYSIBM.SYSPARMS (that is, @SYSTABLES, @SYSCOLUMNS, @SYSROUTINES and @SYSPARMS). For more information on restricting access to existing DB2 tables, refer to *TIBCO Object Service Broker Managing Security*.

Considering Fail Safe Processing

To guarantee consistency when updating both TDS and DB2 data from a single instance of the Gateway in a single transaction, you must use Fail Safe level-1 processing. For more information, refer to Implementing Fail Safe Processing on page 53.

Specifying Gateway Parameters

You must specify gateway parameters on the EXEC statement in the startup JCL, in a data set or in both; however, you must specify required gateway startup parameters using the HDB2SRVC data set.

If you specify parameters in both the EXEC statement and a data set, EXEC statement parameters override data set parameters.

Include required startup parameters in any order, one per record, beginning in column one, and ending with a blank or a comma. An asterisk (*) in column one indicates a comment record.

Defining Parameters in a Data Set

The data set containing startup parameters must be defined as follows:

- DDname HDB2SRVC
- Allocated FB LRECL=80

Sample parameters are provided in member XDB2SRVC of the <HLQNONV>.<INSTVER>.CNTL data set.

Gateway Input Parameters

See Gateway Parameters on page 42 for a list of the DB2 gateway parameters and their default values.

Startup Prerequisites

Prerequisites

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

- If the Data Object Broker is on z/OS and Dynamic Resource Creation is not permitted (Data Object Broker Parameter `DYNAMICRESOURCE = N`), the Gateway must be identified to the Data Object Broker in a permanent resource. To do this, define Gateway resources to the Data Object Broker's resource management repository file.

  If the Data Object Broker is on z/OS and Dynamic Resource Creation is permitted (Data Object Broker Parameter `DYNAMICRESOURCE = Y`) and there is no matching permanent resource, a dynamic resource entry matching the Gateways requirements will be created when the Gateway connects to the Data Object Broker. If there is a permanent entry for the Gateway, it must match the requirements for the Gateway. It is recommended that all permanent entries for DB2 Gateways are deleted from the repository when dynamic resource creation is permitted.

- If the Data Object Broker is on Windows or Solaris, set up National Language Support, if necessary. Refer to *TIBCO Object Service Broker National Language Support* for setup and configuration information.

Default Resource Settings (z/OS with DYNAMICRESOURCE=N only)

Use the Resource Management option (option 3) available from the Administration control group of the TIBCO Object Service Broker Administration Menu. You need to use the Resource Details (PF5) and the Resource Schedules (PF10) screens to specify the connection attributes. To get to the Resource Detail screen you must first specify a type (SERVERTYPE) and group (SERVERID) on the Resource Type screen.

The following table illustrates the attributes for SERVERID=DEFAULT.

Resource Details			Resource Schedule
INTERMEDIATE ROLLBK	EARLY RELEASE	COMMIT LEVEL	ONLINE ONLY
Y	Y	0	N

The default settings of Resource Detail fields are affected by these gateway parameters:

- RESPONSEMODE
- FSLEVEL
- USERTYPE

Depending on the values you set in Specifying Gateway Parameters on page 29, you must specify the defaults as follows:

Parameter Setting	Resource Setting
RESPONSEMODE=SYNC	EARLY RELEASE must be set to N.
FSLEVEL=1	COMMIT LEVEL must be set to 1.
USERTYPE=ONLINE	ONLINE ONLY must be set to Y.

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

Starting the Gateway

When you start the Gateway, the S6BDB2M module connects to both DB2 and TIBCO Object Service Broker. The number of Gateways specified in the SERVERS parameter are then attached by S6BDB2M. Each Gateway (the S6BDB2S module) connects to TIBCO Object Service Broker. When the Execution Environment requests access to DB2 data, a thread to DB2 is established. For more information, refer to Adding Gateway Threads on page 55.

You can run the Gateway as a batch job or as a started task.

Running the Gateway as a Batch Job

A batch JCL is located in member DB2BATCH of the JCL data set.

Running the Gateway as a Started Task

Modify JCL member DB2BATCH in the JCL data set to be a JCL procedure and move it to your site PROCLIB (for example, SYS1.PROCLIB). You can then start the Gateway from the z/OS operator console using the z/OS **START** command.

Region size *must* be large enough to hold all the Static SQL handlers to be used during the life of the Gateway. All Static SQL handlers remain in memory during the life of the Gateway for improved performance.

See Also

TIBCO Object Service Broker Messages With Identifiers for information on messages produced by the Gateway.

TIBCO Object Service Broker for z/OS Installing and Operating for more information on running the Gateway as a batch job.

Shutting Down the Gateway

You can shut down a Gateway using the **MODIFY** operator command from the z/OS operator console, or by using the RESOURCE MANAGEMENT option from the Administration menu.

Using the MODIFY Operator Command

The format of the **MODIFY** command is as follows:

MODIFY *dob_jobname*,**STOPSERVER**=*idprefix*

dob_jobname	The name of the batch job or the system task name under which the Data Object Broker is running.
idprefix	Unique name of the Gateway. The Gateway creates this name by appending a three-digit number to the IDPREFIX parameter specified in the Gateway startup JCL. The default is DB2. To view unique names assigned to existing Gateways, use the RESOURCE MANAGEMENT option from the Administration menu.

Additional MODIFY Commands

You can also use one of three variations of the **MODIFY** operator command to shut down groups of Gateways:

- Shut down all Gateways:

  MODIFY *dob_jobname*,**STOPSERVER=ALLDB2**

- Shut down all Gateways with a common IDPREFIX:

  MODIFY *dob_jobname*,**STOPSERVER=***idprefix**

- Shut down all Gateways with a common SERVERID:

  MODIFY *dob_jobname*,**STOPSERVER=SRVID***serverid*

RESOURCE MANAGEMENT Option

Use the RESOURCE MANAGEMENT option from the Administration menu to shut down either one Gateway or groups of Gateways. For details on the Administration menu, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

Installation Verification

Installation verification for an external DBMS provides a quick method to verify that the installation of a TIBCO Object Service Broker DOB and one or more DBMS Service Gateways was successful. This verifies that the communication between the DOB and a Service Gateway, and a Service Gateway and the DBMS, is functioning properly.

The verification procedure is split between two elements. First, the IVPDB2P member that was placed in `<HLQNONV>.<INSTVER>.JCL` by the installation procedure for the Service Gateway for DB2, binds a plan ($PLNAME$), grants access to it, and starts the Gateway. You may have to change the plan name used in STEP2 depending on your version of DB2. The second element is a batch file, shell script, or JCL member, that is run in the environment of the Data Object Broker, and which accesses data in the external database to verify connectivity.

Each of these steps can be done manually, meaning you can start the Service Gateway for DB2 normally rather than by using IVPDB2P, and you can browse data in the external database via the Gateway without using the shell script/batch file/JCL.

TIBCO Object Service Broker tables are pre-defined with definitions that correspond to the sample tables or demo databases commonly included with the various external DBMSs. In the case of DB2, the table EMP is used. The TIBCO Object Service Broker tables are prefixed with the name `S6BIVP*`, for example `S6BIVP_DB2`.

If a DBMS does not have sample tables or a demo database, or these were not included in its installation, you need to manually verify access to one of your databases. Instructions to perform this can be found later in this manual.

Requirements

Installation verification requires the following:

- The complete installation of a TIBCO Object Service Broker DOB on z/OS, Windows or UNIX.
- The installation of a Service Gateway that must be run from an APF authorized library. If the DOB is running on z/OS, the Gateway must be properly configured in Resource Management.
- Sample tables or demo databases, often bundled with a DBMS and installed at most shops.

- The configuration of the communications path between the Service Gateway and the Data Object Broker.

Before you run the Installation Verification Procedure (IVP), you must configure the communication path that the Service Gateway will use to communicate with the Data Object Broker. The details of this configuration depend upon whether the Data Object Broker is installed on z/OS or on an open systems platform, and upon the communication protocol to be used.

- If the DOB is on z/OS and you are not using TCP/IP to communicate between it and the Gateway, comment out the DD statement for S6BRELAY in the IVPDB2P JCL member.
- If the DOB is on Windows or UNIX, and the communications protocol is TCP/IP, then the communications configuration for the Service Gateway is determined by the relay file placed in `<HLQNONV>.<SLQ>.RELAYCFG` by the installer. A complete discussion of how to configure the relay file is beyond the scope of this document; for details, see *TIBCO Object Service Broker for z/OS Installing and Operating*.

For the purpose of the IVP only, a simplified file will suffice. The following example illustrates a relay file that configures a connection to a remote DOB on Windows/Unix.

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
   <tcpipparms tcbnum="3" maxtcbsockets="50" />
   <directory>
      <node name="WINDOB">
         <tcpip host="192.168.1.1" port="12000" />
      </node>
   </directory>
</relay>
```

Substitute the name, host, and port for your DOB into this template. They can be found in the `HURON.DIR` file in the database directory of the TIBCO Object Service Broker installation to which you are trying to connect.

Member IVPDB2P in data set `<HLQNONV>.<INSTVER>.JCL` contains the job steps required to prepare the Service Gateway for the verification process. If all requirements are met, customize the JCL and run it. Note that if the Service Gateway has already been configured and started, this step is not necessary.

To customize the JCL, replace each instance of TDS in the JCL with the DOB node name that you placed in the relay file.

IVP Batch File, Shell Script or JCL

The IVP is a batch file, shell script, or JCL member in the environment of the DOB. In each case, the IVP uses the Service Gateway to run the rule S6BIVP_VERIFY. This rule reads the contents of the table S6BIVP_DB2810, which in turn uses the Service Gateway to access DB2. Note that if you are using DB2 V9, you should edit this rule to specify the table S6BIVP_ DB2910 instead.

- On Windows, the batch file `ivpdb2.bat` is in the `bin` directory of your TIBCO Object Service Broker installation. Variables within the file need to be customized; directions for doing so are in the file. The customization includes setting where the results will be placed.
- On Solaris, the (ksh) shell script `ivpdb2.ksh` is in the `bin` directory of your TIBCO Object Service Broker installation. Shell and environment variables within the script need to be customized; directions for doing so are in the script. The customization includes setting where the results will be placed.
- On z/OS the JCL member IVPDB2 will be found in the `<HLQNONV>.<INSTVER>.JCL` data set associated with the DOB. The customization will involve supplying values for TDS, USERID, and PASSWORD. The results will be placed in the job's output, and may be viewed using SDSF.

Verification Process

The steps in the verification process are described below.

Manually, or by using IVPDB2P:

1. Perform any DBMS preparation work, such as BIND PLAN or extracting and loading dictionary definitions.
2. Submit JCL member DB2BATCH to start the Service Gateway for DB2.

At the DOB's node, using `ivpdb2.bat`, or the corresponding shell script or JCL member:

1. Run a TIBCO Object Service Broker batch job to access the predefined DBMS sample table.
2. Check the results. Sample data from the database will be present in the output if the verification procedure succeeded.

Chapter 2 Operating Service Gateway for DB2

This chapter provides information on how to define TIBCO Object Service Broker DB@ tables, to supply startup gateway parameters, and to operate Service Gateway for DB2.

Topics

Defining TIBCO Object Service Broker DB2 Tables

Obtaining DB2 Table Definitions

There are two methods that you can use to obtain DB2 table definitions:

Method	Refer to…
Gateway	Using the Gateway Method, page 38.
Extraction	Using the Extraction Method, page 40

After creating your TIBCO Object Service Broker DB2 table definitions, you can bind them. This process is described in Binding TIBCO Object Service Broker DB2 Table Definitions on page 41.

Using the Gateway Method

To use the Gateway method, you *must* have a Gateway running. The Table Definer gets its DB2 table information from the DB2 catalog tables.

SERVERID Parameter

Use the SERVERID parameter to identify a pool of Gateways that are considered equivalent by the Data Object Broker. Using it provides access to more than one DB2 subsystem from a single Data Object Broker, or access to a single DB2 subsystem with different Gateway plans and different Gateway configurations.

To define TIBCO Object Service Broker DB2 tables, the Table Definer must be able to determine the DB2 subsystem from which to get the DB2 metadata. This is determined by the entry in the **ServerID** field in the DB2 table definition.

Note: If you specify I in the `Dictionary` field, the definer will attempt to use the extracted dictionary information and ignore the value in the SERVERID field. For details, see Using the Extraction Method on page 40.

Viewing the DB2 Tables/Procedures List

When you press PF4 on the Table Definition Screen, a list of DB2 Tables/Procedures appears, as shown below. Depending on the value in the Subtype of the Table Definition Screen, this list presents either the creators and their related tables (Subtype N), or the schemas and their related stored procedures (Subtype P or R) available from the DB2 subsystem via the gateway denoted by ServerID.

```
          List of available DB2 tables in e(X)ternal DB2 Dictionary
  Creator: SYS*
    Table: SQ*
--------------------------------------------------------------------------------
_ SYSIBM
  SQTCOLPRIVILEGES
_ SYSIBM
  SQTCOLUMNS
_ SYSIBM
  SQTFOREIGNKEYS
_ SYSIBM
  SQTPRIMARYKEYS
_ SYSIBM
  SQTPROCEDURECOLS
_ SYSIBM
  SQTPROCEDURES
_ SYSIBM
  SQTSPECIALCOLUMNS
_ SYSIBM
  SQTSTATISTICS
S=Select
 PFKEYS: 3 = SELECT & EXIT   ENTER = REFRESH   12 = EXIT
```

The values of the fields Creator (Schema) and Table (Procedure) on the Table Definition Screen are used as search pattern, with the asterisk (*) and question mark (?) wildcards allowed, and an empty value is considered identical to an asterisk. Accordingly, only those tables (procedures) are shown in the list, whose creator (schema) and name match those search patterns. You can then modify the search patterns displayed on the List Screen and press Enter to see the refreshed list.

If the 'S' line command is used, the definer adds to the Table Definition Screen those columns (parameters) of the respective table (procedure) whose names do not match the DB2 names of the fields already selected on Table Definition Screen.

Using the Extraction Method

By using the extraction method, you can store all DB2 table information in a TDS table. This means you can define TIBCO Object Service Broker DB2 tables without having an instance of the Gateway running. However, the data is static, so you must update it whenever *any* changes are made to new or existing DB2 table definitions.

Defining TIBCO Object Service Broker tables using the extraction method requires that the information about DB2 tables/procedures be copied into the TIBCO Object Service Broker internal dictionary tables.

Populate the @DB2SYSTBLS and @DB2SYSRTNS Tables

You can extract the metadata from DB2 (external dictionary) into a set of tables in TIBCO Object Service Broker (internal dictionary), so the definer can use them at the table definition time. The definer switches between, and transparently uses, the external and internal dictionaries as implied by the value - I or X - of the Dictionary field on the Table Definition Screen.

To extract information about the tables/procedures from a DB2 subsystem, you need a Service Gateway for DB2 to be running and available. Using a TIBCO Object Service Broker Workbench of your choice, run either of the following rules:

```
COPY@DB2TBLS(<sid>,<list of creators>)
COPY@DB2RTNS(sid>,<list of schemas>)
```

to extract information about the tables or the stored procedures, respectively. Both of these rules accept two parameters:

- `<sid>` – the SERVERID of the Gateway to use; if empty/NULL, DEFAULT is assumed.
- `<list of creators/schemas>` – a string of blank-delimited names, for example, 'SYSIBM DEVL7083 ABC'; if empty/NULL, information about the tables/procedures for all available creators/schemas is copied.

If the `<list of creators>` or `<list of schemas>` parameter is empty/NULL, the entire internal dictionary (all tables or all procedures, respectively) is cleaned up. However, if the list contains at least one value, only the tables/procedures pertaining to those values are removed from the internal dictionary. In both cases, the removal takes place prior to the beginning of the copy process. This approach allows you to easily construct a fresh copy of the entire DB2 dictionary or selectively refresh the metadata denoted by one or more particular creators/schemas.

Binding TIBCO Object Service Broker DB2 Table Definitions

You can bind a DB2 table definition but not its data. TIBCO Object Service Broker DB2 tables for which you request binding are bound to both the Execution Environment and the Gateway when they are accessed from a rule.

Rebinding

If you change a definition, it is automatically rebound in the Gateway.

Specifying Maximum Space Available

You can specify the maximum amount of space available to hold all TIBCO Object Service Broker DB2 table definitions by using the POOLSIZE gateway startup parameter. For details, see Gateway Parameters on page 42.

See Also *TIBCO Object Service Broker Application Administion* for information on binding tables.

Gateway Parameters

You specify gateway startup parameters in the HDB2SRVC data set. For more information on specifying gateway parameters, refer to Specifying Gateway Parameters on page 29.

Parameters that you must set include PLAN, SSID, and TDS. You can add or modify the other parameters as required.

BROWSEPLAN	Name of the Gateway plan to be used if the browse flag is on. If omitted, the plan used is the one specified for the PLAN parameter.
DEBUG	Specifies debugging messages should be echoed on the Gateway and placed in the system message log.
DECIMALPOINT	Informs the Gateway how the DB2 environment represents decimal points. Valid values: **COMMA** – The DB2 subsystem uses commas to indicate decimal points. **PERIOD** – Default. The DB2 subsystem uses periods to indicate decimal points.
EXTERNALUSERID	Applicable only if SECLEVEL=1. Specifies how the primary authorization ID is identified to the DB2 subsystem. Valid values: **USERID** – Default. The TIBCO Object Service Broker session ID. **GROUP** – The current TIBCO Object Service Broker security group. This group can be up to 16 characters but only 8 characters are supported. A SECURITYFAIL occurs for more than 8 characters.

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 53 for more information. You must specify the TRXDB, FSTABLENAME, and RECOVERYID 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 53 for more information.
FSTABLENAME	Required only if FSLEVEL=1. Specifies the name of the TIBCO Object Service Broker DB2 table that maps to the DB2 transaction table. **Note**: If you also specify the TRXDB parameter, the system uses TRXDB=*creator.tablename* and replaces the DB2 creator and tablename in the TIBCO Object Service Broker DB2 table specified in FSTABLENAME. If you have multiple TIBCO Object Service Broker DB2 initializer programs with the same SERVERID, the FSLEVEL, FSTABLENAME, TRXDB, and RECOVERYID gateway parameters must have the same values for each initializer program. Default: @DB2FSTRXDB, which points to the DB2 table *creator*.HRNTRXDB.
IDPREFIX	Each instance of the Gateway *must* have a unique IDPREFIX. This parameter is the prefix to be used to construct a unique name for each Gateway address space. The Gateway appends three decimal digits to this prefix and uses the result to log in to TIBCO Object Service Broker. It uniquely identifies each Gateway to TIBCO Object Service Broker. This parameter can have up to five characters. Default: DB2.
MDL	The pattern for selecting the VTAM ACB name that the Gateway uses for communications. Default: the pattern assumed from the model specified for the TDS gateway parameter. For example, if you do not specify MDL and TDS=OSB0001, the Gateway uses the pattern *OSBnnnn*. If you do not specify MDL, ensure that the value of TDS is a valid VTAM ACB model.

NONSWAPPABLE	Specifies whether the Gateway is marked as nonswappable after initialization. The Gateway must be running authorized for this parameter to take effect. Valid values are **YES** (default) and **NO**. The optimum way to run a Gateway is authorized, as a started task, and nonswappable. This way the Gateway uses symmetrical cross memory services.
NOSTAE	S6BDB2M attaches Gateway subtasks (S6BDB2S) without ESTAE EXIT. Include this parameter for abend debugging.
PLAN	The Gateway plan name. The name must correspond to the name used in the bind input member XLINKDB5. Refer also to BROWSEPLAN on page 42 and UPDATEPLAN on page 47. Default: S6BDB2S.
POOLSIZE	The amount of space (in kilobytes) to hold TIBCO Object Service Broker DB2 table definitions in the Gateway. This parameter take a value in the range 32 to 16384. An estimate of the minimum number of TIBCO Object Service Broker DB2 tables that can be accessed in a single transaction is POOLSIZE divided by the maximum CTABLESIZE. Refer to Estimating the CTABLESIZE Parameter on page 49 for more information. Default: 256 (KB).
RECOVERYID	Applicable only if FSLEVEL=1 and SECLEVEL=1. Sets the DB2 primary authorization ID that you want to use during recovery when querying the DB2 transaction table to see if the transaction completed (in-doubt transactions). The ID can be up to eight characters and must have EXECUTE authority to the Gateway plan and SELECT access to the DB2 transaction table. Default: the ID used to start the Gateway. **Note**: If you have multiple TIBCO Object Service Broker DB2 initializer programs with the same SERVERID, ensure that the FSLEVEL, TRXDB, FSTABLENAME, and RECOVERYID gateway parameters are assigned the same values for each initializer program.

RESPONSEMODE	The mode that the Gateway uses to respond to requests. You can use this mode to release a Gateway for a new transaction more quickly in read-only situations. Valid values: **ASYNC** – Default. The Data Object Broker releases the Gateway from the transaction when it sends a transaction end message to the Gateway (it does not wait for transaction end confirmation from the Gateway). **Note** This applies if the Gateway processed only query requests. **SYNC** – Causes the Data Object Broker to wait for the Gateway to complete the end of transaction processing (SQL COMMIT success or failure result from DB2). For use only with a Data Object Broker on z/OS. This parameter corresponds to the EARLY REL Gateway path management field. For more information, refer to Startup Prerequisites on page 30.
SCOPE	The length of time to hold the connection between an instance of the Gateway and a DB2 subsystem. Valid values: **SESSION** – Hold the connection for the life of the Gateway. The connection is not established until the first request for DB2 data is processed by the Gateway. **TRANSACTION** – Hold the connection for the life of a TIBCO Object Service Broker transaction. *nnnn* – Hold the connection for *nnnn* seconds beyond the end of the TIBCO Object Service Broker transaction. Valid entries are 0-9999. Default: if SECLEVEL=0, SESSION; if SECLEVEL=1, TRANSACTION.
SECLEVEL	Specifies the level of authorization to use when accessing DB2 data. Refer to Implementing External Security on page 52 for more information. Valid values: **0** – Default. Access the DB2 subsystem using the ID that started the Gateway. The ID is verified by the DB2 subsystem, not TIBCO Object Service Broker. Do not use an external security interface. **1** – Access the DB2 subsystem based on the user's TIBCO Object Service Broker session ID or the current group name. Use an external security interface. If SECLEVEL=1 and FSLEVEL=1, review the default for the RECOVERYID parameter. If SECLEVEL=1, review the defaults for the EXTERNALUSERID and SCOPE parameters.

SERVERID	Identifies a pool of Gateways with common characteristics such as Gateway plan, POOLSIZE, DB2 subsystem, and the rest. If you have multiple TIBCO Object Service Broker DB2 initializer programs with the same SERVERID, ensure that the FSLEVEL, RECOVERYID, FSTABLENAME, and TRXDB gateway parameters have the same values for each initializer program. You *must* specify a server ID when defining a TIBCO Object Service Broker DB2 table so the definition uses the correct version of SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS. This parameter can be up to eight characters. The SERVERID parameter can be overridden at runtime. Refer to Dynamically Changing Gateway Parameters on page 50 for more information. Default: DEFAULT (uses @SYSCOLUMNS and @SYSTABLES).
SERVERS	The number of Gateway tasks that the TIBCO Object Service Broker DB2 initializer program should attach to the Gateway address space at startup. This number can be from one to a value less than or equal to the value set in the **Maximum Connection Count** field in your network configuration. Default: 1.
SSID	The DB2 subsystem with which the Gateway communicates. This parameter can have up to four characters.
STATICCURSOR SELECT	Specifies the method of SQL creation. Specify STATICCURSORSELECT=EXACT or omit the parameter. EXACT forces dynamic SQL creation if an exact static handler cannot be found.
TDS	The Communications Identifier of the Data Object Broker with which the Gateway communicates.

TRACE	Invokes a debugging tool showing SQL errors and warnings, and SQL statements as they are passed to DB2 (for SELECT, INSERT, UPDATE, and DELETE processing). OPEN CURSOR, CLOSE CURSOR, and FETCH are not logged. To activate TRACE you must also: • Allocate an FB LRECL=97 data set. • Add a DD statement to the Gateway JCL with DDNAME HDBTRACE. The trace is formatted in columns as follows: **1 - 80**: The statement or error. Also shows the static handler load module name if Static SQL is used, and the SQL statements and value of any variables if Dynamic SQL is used. **81**: The continuation marker. **82 - 89**: The ID of the Gateway (IDPREFIX followed by Gateway number). **90 - 97**: The TIBCO Object Service Broker transaction ID (use with the DEBUG parameter).
TRXDB	The transaction database. Required only if FSLEVEL=1 and if the actual Fail Safe DB2 transaction table name and/or creator are different from the ones to which the TIBCO Object Service Broker DB2 table name specified using the FSTABLENAME parameter points. The name of the DB2 transaction file to contain information about the last transaction processed by the Gateways. **Note**: Alternatively, you can specify the FSTABLENAME parameter. If you have multiple TIBCO Object Service Broker DB2 initializer programs with the same SERVERID, the FSLEVEL, FSTABLENAME, and TRXDB gateway parameters must have the same values for each initializer program.
UPDATEPLAN	The name of the Gateway plan to be used if the browse flag is off. If omitted, the plan used is the one specified for the PLAN parameter.
USEDB2LIKE	Include this parameter if DB2 LIKE statements are to be generated instead of TIBCO Object Service Broker LIKE statements when using FORALL…LIKE statements to select data for evaluation. For more information, refer to FORALL Statement on page 110.

USERTYPE	The type of user allowed to be connected to this Gateway. Valid values: • **ANY** – Default. Indicates that batch and online users can use the Gateway. • **ONLINE** – Reserves the Gateway for online users only. When allocating a Gateway, the Data Object Broker first tries to allocate an online user to an online Gateway. If it cannot, it tries to allocate the user to a Gateway with USERTYPE=ANY. For use only with a Data Object Broker on z/OS. This parameter corresponds to the **ONLINE ONLY** Gateway path management field. For more information, refer to Startup Prerequisites on page 30.

Estimating the CTABLESIZE Parameter

When you select DB2 columns as TIBCO Object Service Broker DB2 fields, the number of fields that you can access using a TIBCO Object Service Broker DB2 table definition is dependent upon the CTABLESIZE Data Object Broker parameter. To estimate the number of bytes required to support a specified number of fields, execute the following rule:

```
ESTIMATETBLDFN(NUM_FIELDS)
```

You must supply one argument, which is the maximum number of fields accessed by a TIBCO Object Service Broker DB2 table in your system. The rule returns an estimate of the maximum CTABLESIZE required (for each TIBCO Object Service Broker table type) to support this number of fields.

Result of Executing ESTIMATETBLDEFN for 50 Fields

```
------------------------------ INFORMATION LOG -------------------------------
COMMAND ===>                                                  SCROLL ===> P
DATE: Nov 28,2006      REPORT ON ESTIMATE CTABLESIZE
                         FOR "50" FIELDS

Table Type       CTablesize(K)      XTablesize(K)
----------       -------------      -------------

ADA                 5
DAT                 7
DB2                 5
IDM                 6
IMS                 6                  3
MAP                 4
SLK                 4
204                 6
TDS                 3



PFKEYS: 2=NEXT LOG 3=EXIT 5=REPEAT FIND 12=EXIT 13=PRINT 9=RECALL
```

See Also

- *TIBCO Object Service Broker for z/OS Installing and Operating* for information on TIBCO Object Service Broker network configuration.
- Refer to *TIBCO Object Service Broker Parameters* for information about the CTABLESIZE Data Object Broker parameter.

Dynamically Changing Gateway Parameters

When a table is defined, attributes specific to external DBMS table types are held in the @SERVERPARMS control table, which is parameterized by table name. Each occurrence in the table specifies a value, such as SERVERID and SERVERTYPE, related to the external environment. The complete set of such attributes for a particular table type can be found in the @@SERVERPARMS control table parameterized by table type.

Sample @@SERVERPARMS Control Table for Table Type DB2

```
EDITING TABLE    :   @@SERVERPARMS(DB2)
COMMAND ==>
                                                                SCROLL: P
    NUMBER          NAME        TYPE  SYNTAX  LENGTH  DECIMAL     DEFAULT
_ ----------- ----------------  -     -      ------  -------  ----------------
_           1 SERVERID           S     C           8        0  DEFAULT
_           2 SERVERTYPE         S     C           3        0
_           3 CODEPAGE           S     C          32        0
_           4 VERSION            Q     V           8        0  0
_           6 DB2CREATOR         S     V         128        0
_           7 DB2NAME            S     V         128        0
_           8 DB2LOCATION        S     C          16        0
_           9 STATICSQL          S     C           1        0  N
_          10 STATICSQLHANDLER   S     C           8        0


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

Using SETXPARM and RESETXPARM

At runtime, you can dynamically modify the SERVERID gateway startup parameter using the SETXPARM and RESETXPARM shareable tools. This reduces the number of table definitions required to define the external data. The changes are stored in either of two session tables:

Table	Function
@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 session, until SETXPARM is invoked again, or the overrides are reset.

Gateway Parameters that Can Be Overridden at Runtime

The gateway parameters that can be dynamically changed with SETXPARM and RESETXPARM are SERVERID and SERVERTYPE. The value for SERVERTYPE is DB2.

Examples Using SETXPARM and RESETXPARM

- This example sets the SERVERID for all tables of type DB2 to TORONTO:

```
CALL SETXPARM('TABLETYPE', 'DB2', 'SERVERID', 'TORONTO', '');
```

- This example resets the SERVERID for tables of type DB2 to the Table Definer default value:

```
CALL RESETXPARM ('TABLETYPE', 'DB2', 'SERVERID', '');
```

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

Operations

Implementing External Security

External security is used only if SECLEVEL=1. The IBM-supplied exit DSN3@ATH is invoked each time a connection to DB2 is requested. Refer to the SCOPE parameter in Gateway Parameters on page 42 for information on the duration of connections.

Security Process

DSN3@ATH passes control to the HRNSECD2 macro supplied by TIBCO Object Service Broker. HRNSECD2 performs the following checks:

- Verifies that the DB2 connection requestor is the S6BDB2S Gateway program.
- Verifies that the user ID requesting to start the Gateway is authorized to do so. The table of authorized users is in the HRNSECDT macro.
- Verifies that the DB2 connection requestor is not a privileged DB2 user ID (that is, SYSADM). The table of unauthorized users is in the HRNSECDZ macro.

If the DB2 connection requestor passes all the previous checks, HRNSECD2 then changes the DB2 primary authorization ID to the TIBCO Object Service Broker session ID or to the current group name depending on the setting of the EXTERNALUSERID gateway parameter. If the first of these checks fails, HRNSECD2 returns control to DSN3@ATH with no changes. If one of the remaining checks fail, a SECURITYFAIL exception is raised.

Secondary Authorization ID Processing

Depending on the external security interface you are using, the DSN3@ATH exit sets secondary authorization IDs as follows:

CA-ACF2	Sets the primary authorization ID and calls CA-ACF2 to acquire secondary authorization ID information for each thread connection.
RACF	Assumes that the primary authorization ID used to start an address space is the DB2 primary authorization ID and that its associated secondary authorization IDs are used for all thread connections.

The external security interface changes *only* the primary authorization ID to the DB2 connection requestor; it does *not* remove the secondary authorization IDs from the security control block built by DSN3@ATH for RACF. Ensure that the RACF user ID that starts the Gateway does not have any secondary authorization IDs associated with it. This ID also does not require any DB2 security authorization.

If you require secondary authorization ID processing to implement RACF group checking, simulate this process as follows:

1. Create the equivalent TIBCO Object Service Broker group.
2. Set the EXTERNALUSERID gateway parameter to GROUP.
3. If Fail Safe level 1 is requested, ensure that the RECOVERYID gateway parameter is set to an appropriate authorization ID since the ID that starts the Gateway is the default recovery ID.

Installing the External Security Interface

Depending on which external security interface you are using, the sample source for the DSN3@ATH exit is located as follows:

CA-ACF2	In the ACFMAC data set, member name ACF3@ATH.
RACF	In the DSNSAMP data set, member name DSN3SATH.

The HRNSECD2, HRNSECDT, and HRNSECDZ TIBCO Object Service Broker macros are located in the MACRO data set distributed with TIBCO Object Service Broker. Installation instructions for each macro are detailed in the header information.

Implementing Fail Safe Processing

Fail Safe level-1 processing guarantees consistency when updating both TDS and DB2 data from a single Gateway in a single transaction. At the end of a transaction, the Data Object Broker requests that the Gateway commit outstanding updates. As part of DB2 commit processing, the Gateway updates a DB2 transaction table to record the successful commit. If the Gateway does not respond to the Data Object Broker in a reasonable amount of time, the transaction is flagged as being in doubt. Locks held on TDS data remain until the problem is resolved.

When a connection is re-established between the Data Object Broker and an instance of the 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 DB2 transaction table to determine this. If the update was completed in DB2, the TDS updates are applied and the locks are released.

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

See Also Refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for more information on Fail Safe processing.

Procedural Overview

To implement Fail Safe Processing, you must complete the following tasks:

1. Define a DB2 transaction table, page 54
2. Define a TIBCO Object Service Broker DB2 transaction table, page 54
3. Modify the Gateway Fail Safe startup parameters, page 54

These tasks are described in the following sections.

Task A Define a DB2 transaction table

To implement Fail Safe level-1 processing, you must define the DB2 table found in the CNTL data set in member XDB2TRXD. Do not change the DB2 Column definitions. This transaction table holds a maximum of one record for each combination of the Gateway and Data Object Broker.

Task B Define a TIBCO Object Service Broker DB2 transaction table

After defining a DB2 transaction table you must define a TIBCO Object Service Broker DB2 transaction table that points to the DB2 transaction table. A sample of this table is @DB2FSTRXDB, which uses server ID DEFAULT and points to the DB2 table *creator*.HRNTRXDB.

Task C Modify the Gateway Fail Safe startup parameters

Ensure that the following gateway startup parameters are included in your Gateway startup JCL:

- FSLEVEL=1

- One of the following:
  - FSTABLENAME=@DB2FSTRXDB
  - TRXDB=*creator.tablename*

For more information on these startup parameters, see Gateway Parameters on page 42.

At startup, the Gateway asks the Data Object Broker for the TIBCO Object Service Broker DB2 transaction table definition specified in the FSTABLENAME gateway startup parameter. This table definition is bound for the life of the Gateway and is used at transaction end to update the DB2 transaction ID database.

You can manage the transaction table in TIBCO Object Service Broker like any other DB2 table. For example, you can write a TIBCO Object Service Broker rule to clean up the Fail Safe database when shutting down the Gateway.

It is possible that the default name of the TIBCO Object Service Broker DB2 transaction table (@DB2FSTRXDB) changed when you defined it in Task B, Define a TIBCO Object Service Broker DB2 transaction table, on page 54.

Other Operational Procedures

Adding Gateway Threads

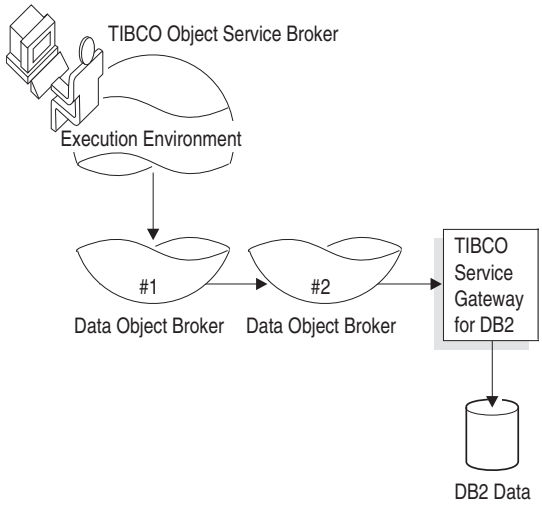
The number of Gateways attached to the Gateway address space is specified in the Gateway startup JCL. You can use either of the following methods to add additional Gateways:

- Shut down the Gateway and start it again with an increased number of Gateway (using the SERVERS gateway parameter).
- Start another instance of the Gateway with the same SERVERID and a different IDPREFIX.

Using Distributed Data with the Gateway

Distributed access between TIBCO Object Service Broker and DB2 is allowed subject to requirements of all distributed access. Refer to *TIBCO Object Service Broker Application Administion* and *TIBCO Object Service Broker for z/OS Installing and Operating* for information on distributed access.

This illustration shows a sample Gateway distributed-data scenario:



Displaying the Status of instances of the Gateway

Use the RESOURCE MANAGEMENT option from the Administration menu to display the status of an instance of the Gateway. The sample screen below shows the type of information displayed for resource type DB2.

```
S6BADM33  HTSTSRV       RESOURCE DETAIL FOR DB2 DB2SRV        2000JAN02 16:40:52

INTERMEDIATE ROLLBK  N EARLY RELEASE     Y LAST USER REUSE     N COMMIT LEVEL  0
RETRY INTERVAL       0 TP NAME             USER ID PREFIX       FAILURES       0
NODE  MMMSYSTEMA01     INDOUBTS          N                      DELETE

                      CONNECTIONS      IN-USE         TRX          MESSAGE
                     CUR   MAX   LMT   CUR   MAX      COUNT          COUNT
          ONLINE       0     0     0     0     0          0              0
          COMMON       0     0    25     0     0          0              0

SCHEDULE NAME IMSIMSDRASV

         APPLICABLE DAYS          EXCEPTION  START  ONLINE  CONNECTIONS
    MON TUE WED THR FRI SAT SUN   DATE        TIME   ONLY       MAX
     Y   Y   Y   Y   Y   Y                   00:00    N          25



 ENTER-PATHS PF2-TYPE PF4-GROUP PF5-PEER PF9-START PF10-SCHEDULES PF11-UPDATE
THERE ARE NO ACTIVE PATHS TO BE DISPLAYED, REQUEST IGNORED
```

Debugging Rules and Applications

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

To determine the SQL statements being sent to DB2 by your rules, include the TRACE parameter in your Gateway startup JCL. The SQL statements for all DB2 table accesses are included in the trace. To view only SQL statements created by your rules, you require a dedicated Gateway. This can be accomplished using the SERVERID gateway startup parameter. To change the server ID of a TIBCO Object Service Broker DB2 table definition to an alternate dedicated server ID, execute the CHANGE_SERVERID tool as follows:

```
CHANGE_SERVERID(table_name,old_serverid,new_serverid)
```

When the trace shows the application is using Static SQL, it displays the handler load module and cursor name. You must review the Static SQL handler assembler code to see the SQL statements associated with the named cursor.

Reporting Problems

Refer to How to Contact TIBCO Support on page xvi for information about reporting problems. Have the following information available when reporting the Gateway related problems to TIBCO Support:

- TIBCO Object Service Broker DB2 table definitions and sample data
- Listings of all the TIBCO Object Service Broker DB2 control tables
- The Gateway job log and control cards
- Output from the Gateway debug trace (if applicable)

See Also *TIBCO Object Service Broker Application Administion* and *TIBCO Object Service Broker for z/OS Installing and Operating* for more information on distributed access.

TIBCO Object Service Broker for z/OS Installing and Operating for more information on the RESOURCE MANAGEMENT option.

TIBCO Object Service Broker Programming in Rules for more information on debugging rules and applications.

Connecting the Gateway to a Windows, or Solaris Data Object Broker

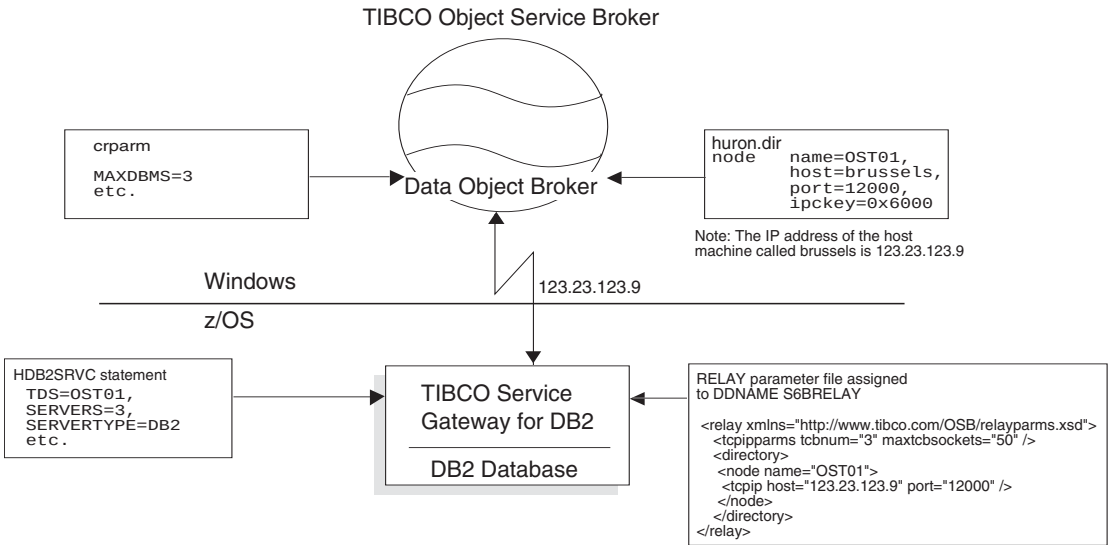
You can configure the Data Object Broker and the Gateway to reside on different domains and operating systems (z/OS, Windows, or Solaris). The Gateway must be in the same domain as the DB2 database system.

The following configuration tasks are required to access a Data Object Broker from a different operating environment than your Gateway:

1. Configure the TCP/IP connection on z/OS, page 58
2. Configure the TIBCO Object Service Broker TCP/IP environment, page 59
3. Specify the number of Gateways connecting to the Data Object Broker, page 59
4. Specify the gateway parameter, page 60

Sample Configuration

The following diagram shows a sample configuration:



Task A Configure the TCP/IP connection on z/OS

Prepare the TIBCO Object Service Broker relay file – RELAYCFG member in the CNTL data set. This file associates the TIBCO Object Service Broker communications identifier with the TCP/IP application addressing information.

Sample Relay File Assigned to DDNAME S6BRELAY

```
<relay xmlns="http://www.tibco.com/OSB/relayparms.xsd">
    <tcpipparms tcbnum="3" maxtcbsockets="50" />
    <directory>
     <node name="OST01">
      <tcpip host="123.23.123.9" port="12000" />
     </node>
    </directory>
</relay>
```

The element and attribute names in the relay file are case sensitive.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for detailed information about preparing the TIBCO Object Service Broker relay file.

TIBCO Object Service Broker Parameters for details about the parameters used in the relay file and how to specify them.

Task B Configure the TIBCO Object Service Broker TCP/IP environment

Add the following parameters for the TCP/IP connection to the Data Object Broker directory file, huron.dir:

name	This must be the same value as the node name set in the RELAY parameter file described in Task A, Configure the TCP/IP connection on z/OS, on page 58.
host	The name of the host machine where the TIBCO Object Service Broker monitor process listens for connections.
port	The number of the TIBCO Object Service Broker monitor socket port.
ipckey	The value of the IPC key.

Task C Specify the number of Gateways connecting to the Data Object

Broker

Specify the following value for the MAXDBMS parameter in the crparm file for your Data Object Broker.

MAXDBMS	This must be equal to or greater than the value specified in the SERVERS parameter in the startup JCL or HDB2SRVC SYSIN file.

Task D Specify the gateway parameter

Specify the following value for the TDS parameter in the HDB2SRVC statement in the startup JCL or HDB2SRVC SYSIN file. See Gateway Parameters on page 42 for more details about specifying gateway parameters.

TDS	This must be the same value as the node name set in the relay file described in Task A, Configure the TCP/IP connection on z/OS, on page 58.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for detailed information about preparing the relay file.

TIBCO Object Service Broker Parameters for details about the parameters used in the relay file and how to specify them.

Chapter 3 Using Static SQL

This chapter shows you how to use static SQL.

Topics

Why Use Static SQL?

Using Static SQL for your application enables DB2 accesses to be performed more quickly. One Static SQL handler is created for each TIBCO Object Service Broker DB2 table so that the Table Browser, the Table Editor, or any new application can take advantage of existing Static SQL for an individual TIBCO Object Service Broker DB2 table, rather than for an application.

Limitations to Static SQL

Static SQL is less suited to a test environment, because you must do the entire procedure each time you add new DB2 accesses to your application or change the `Orders` field in the TIBCO Object Service Broker DB2 table definition. If you modify another field in the definition of a TIBCO Object Service Broker DB2 table, you must repeat only Steps 3 through 8 of the following procedure.

Static SQL handlers cannot be generated if your CREATOR.TABLE name is greater than 44 characters long.

Procedure Overview

To define Static SQL, complete the following tasks:

1. Task A: Log DB2 Accesses, page 63
2. Task B: Prepare to Generate Static SQL, page 64
3. Task C: Generate Static SQL Using the @STATICSQL Tool, page 65
4. Task D: Manage Warning and Error Messages from @STATICSQL, page 70
5. Task E: Assemble the Static SQL Handler, page 73
6. Task F: Link the Static SQL Handler, page 74
7. Task G: Bind the DB2 Gateway Plan with Static SQL, page 75
8. Task H: Update the Gateway Startup JCL, page 75

These tasks are described in the following sections.

Task A: Log DB2 Accesses

When you complete a TIBCO Object Service Broker application, you must log the TIBCO Object Service Broker DB2 access statements to generate Static SQL.

Procedure

Ensure that you have an Execution Environment available with the DB2LOG Execution Environment parameter set to Y. To collect TIBCO Object Service Broker DB2 access statements, run the application using this Execution Environment. Accesses are collected in session tables while the application is running, and written to TDS tables at logoff time. Refer to , Collecting TIBCO Object Service Broker DB2 Access Statements, on page 121 for more information.

Only one Static SQL handler is created for each TIBCO Object Service Broker DB2 table. If a table is accessed by more than one application, all these applications must be run. If you modify your application, you *must* rerun the application, or only the portion of the application that changed, to collect any new TIBCO Object Service Broker DB2 access statements.

If an access to a TIBCO Object Service Broker DB2 table was already logged, it is not logged again.

See Also *TIBCO Object Service Broker Parameters* for further information on parameters.

Task B: Prepare to Generate Static SQL

Edit Import and Export Table Definitions

Before you execute @STATICSQL for the first time, you must edit the definitions for the @DB2SERVERJCLMDL, @DB2SERVERASM, and @DB2SERVERJCL import and export tables. Initially, only level-7 TIBCO Object Service Broker users can edit these tables; if a level-1 user needs to customize these tables, you must update the security.

Customization

Customize these tables as follows:

Import table @DB2SERVERJCLMDL	In the `File` field on the Table Definition screen, specify the name of the JCL data set containing members ASMDB2MD and LINKD2MD. These are models to generate JCL. You must customize these members to specify the required DB2 data sets.
Export table @DB2SERVERASM	In the `File` field of the Table Definition screen, specify the name of the data set that holds the Static SQL handlers generated by @STATICSQL.
Export table @DB2SERVERJCL	In the `File` field of the Table Definition screen, specify the name of the data set that holds the JCL, link control cards, and bind control cards generated by @STATICSQL.

Ensure these data sets are allocated as part of the installation procedure. Also ensure that @SCHEDULEMODEL('MVS', '*DEFAULT*'), used to generate Static SQL in batch, and the BINDDB2S member in the JCL data set are customized. Refer to Requirements for Accessing DB2 Using Static SQL on page 24 for more information.

See Also

- *TIBCO Object Service Broker Managing Security* for more information on updating Security accesses.
- *TIBCO Service Gateway for Files Installing and Operating* about modifying import and export tables.

Task C: Generate Static SQL Using the @STATICSQL Tool

@STATICSQL Functionality

@STATICSQL generates members containing the Static SQL accesses and members used to assemble, link, and assist in binding the Static SQL handlers. This requires access to the DB2 table definition. If you used the extraction method to create the TIBCO Object Service Broker DB2 table definition (refer to Using the Extraction Method on page 40), ensure that your DB2 definitions are current. If you used the Gateway method to create the TIBCO Object Service Broker DB2 table definition (refer to Using the Gateway Method on page 38) you must have an instance of the Gateway running that has the same server ID as your TIBCO Object Service Broker DB2 table definition.

If you modify the TIBCO Object Service Broker DB2 table definition, you have to regenerate only your Static SQL to pick up the changes. You do not have to recollect the TIBCO Object Service Broker DB2 access statements, unless changes were made to your application.

Using DB2 LIKE or NOT

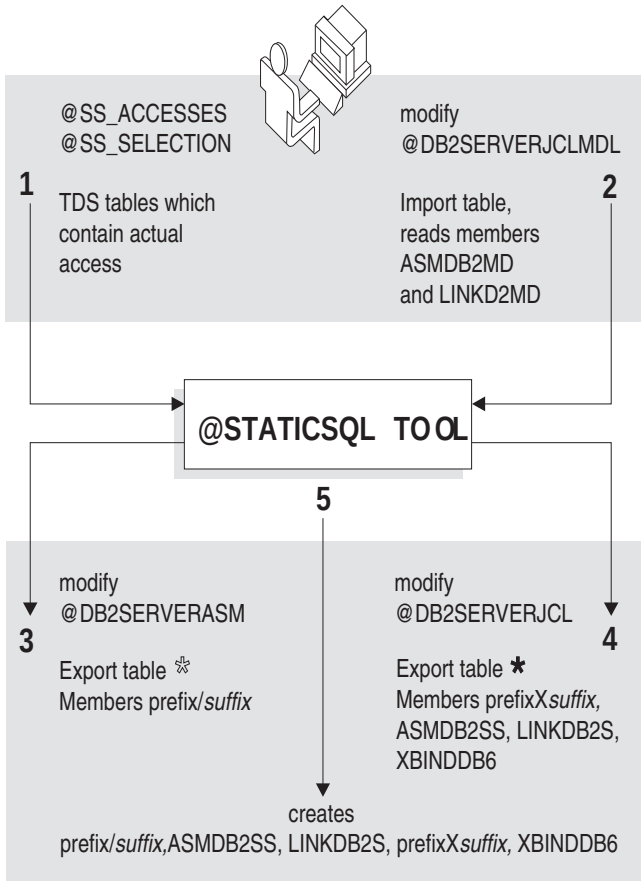
To use DB2 LIKE in your static handlers, you must specify USEDB2LIKE=Y in your Execution Environment start up. By default, LIKE statements are not included in the selection logged.

Using DB2 LIKE

To use the DB2 LIKE statement format (not to be confused with TIBCO Object Service Broker LIKE) you must apply both of the following settings:

- For dynamic SQL, to make the DB2 Gateway formulate DB2 LIKE clauses you must specify USEDB2LIKE in the DB2 Gateway start up parameters. Without this specification, the Gateway parses the rows returned.
- When generating Static SQL using the @STATICSQL tool, to have LIKE statements included in the Static handler you must sign on to the Execution Environment using USEDB2LIKE(Y). LIKE statements are generated in the static SQL handler. Without this specification, LIKE statements are not included in the static handlers.

Process Used to Generate Static SQL.



* Member ASMDB2SS assembles prefix/*suffix*
Member LINKDB2 lnks the Static SQL using member prefixX*suffix*
Member XBINDDB6 is used when binding the TIBCO Service Gateway for DB2 server plan

Static SQL Screen

To invoke the @STATICSQL tool, execute the @STATICSQL rule from the workbench. A screen similar to the one following displays:

```
               Defining Static SQL for use by DB2 Server
Assemble modules generated     Link Static SQL Handlers     Re-Bind DB2 Server
Catalog ServerID: DEFAULT
                             Handler
   Metadata Table Status Pre/Suffix DB2 Table
   ---------------   -   --- -----  ---------------------------------------------
_ @DB2FSTRXDB                       PYM80
                                    FSTRXDB_D529731@

_ @SYSCOLUMNS                       SYSIBM
                                    SYSCOLUMNS

_ @SYSDATABASE                      SYSIBM
                                    SYSDATABASE

_ @SYSPARMS                         SYSIBM
                                    SYSPARMS

_ @SYSROUTINES                      SYSIBM
                                    SYSROUTINES

_ @SYSSTOGROUP                      SYSIBM
                                    SYSSTOGROUP

G=GENERATE   R=REMOVE
PFKEYS: 1=HELP 3=GENERATE(ONLINE) 15=GENERATE(BATCH) 12=CANCEL
```

Static SQL Screen Fields

Metadata Table	Displays all the TIBCO Object Service Broker DB2 tables to which you have READ_DEFN access. You cannot change this name; it is assigned from the Table Definer.
DB2 Table	Displays the name of the DB2 table corresponding to the TIBCO Object Service Broker DB2 table name displayed in the MetaStor Table field of the same row. You cannot change this name; it is assigned from the Table Definer.

Status	The current status of each table. Valid values: **G** – Static SQL was generated for the table. **I** – Static SQL was generated and the definition was changed (invalidated). You must regenerate Static SQL for this table. **blank** – No Static SQL was generated.
Pre/	Type a three character prefix that is unique within all TIBCO Object Service Broker DB2 tables in your system. This prefix is used in conjunction with the suffix.
Suffix	Type a five character suffix that is unique within all TIBCO Object Service Broker DB2 tables in your system. This suffix is appended to the prefix to create the member names of the Static SQL handlers and link control cards. You can edit this field when you regenerate Static SQL. If you have already generated Static SQL for a table, the suffix you entered when you generated Static SQL for this table appears.

Line Commands

The following line commands are available from the Defining Static SQL screen:

R Remove Static SQL for the table (press PF3 to remove it online or PF15 to remove it in batch mode). This command performs the following:

- Resets the table definition, so the Gateway does not use Static SQL for the table
- Removes references to the Static SQL handler in the JCL members ASMDB2SS and LINKDB2S and CNTL bind member XBINDDB6
- Removes occurrence for this table from the @SS_GENERATED table

Note You can re-bind the Gateway plan to remove the Static SQL handler; however, the Gateway uses dynamic SQL irrelevant of the plan.

You can perform the following cleanup functions at a later time:

- Delete previously generated code in data set holding the Static SQL handlers generated by @STATICSQL (file specified in @DB2SERVERASM)
- Delete the Static SQL handler member from the object code, DBRM, and LOAD data sets, and the data set containing the link control cards (file specified in @DB2SERVERJCL)

G Generate Static SQL for the table (press PF3 to generate it online or PF15 to generate in batch mode). This command does the following:

- Creates member *prefix/suffix* in the file pointed to by the export table @DB2SERVERASM. This member is the assembler code that contains the Static SQL statements.

  WARNING Do not modify this member.

- Creates JCL members ASMDB2SS and LINKDB2S, link control card member *prefix/suffix*, and CNTL bind member XBINDDB6 pointed to by the export table @DB2SERVERJCL. Use PF3 to exit, with the following message:

  ```
  Generate/Remove Static SQL: # errors. CHECK LOG for warnings
  ```

  Note Always use PF2 to review warning or error messages.

  Use PF15 to exit, with the following message:

  ```
  Batchjob submitted for:xxxx time:hh:mm:ss to generate static sql
  ```

  Note Always review the message log displayed at the end of the batch job for warning or error messages.

Task D: Manage Warning and Error Messages from @STATICSQL

@SS_SELECTION Table

The screen below shows a sample TIBCO Object Service Broker @SS_SELECTION table that is used to generate Static SQL.

```
BROWSING TABLE   :  @SS_SELECTION(U40_DB2_EMP)
COMMAND ==>
                                                                SCROLL: P
      KEY      ACCESS INCLUDE CURSORID
_ -----------   -       -       -
_           1   N               .
_           4   N               .
_           5   N               .
_           6   N               .
_           7   N               .
_           8   G               .
_           9   N               .
_          10   N               .
_          11   N               .




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

Field Values

KEY	A unique identifier for the access statement. This number is used when building the cursor name.
ACCESS	Identifies the type of access statement: **G** – GET **N** – FORALL

INCLUDE	Identifies if the access statement should be included when you generate Static SQL: **blank** – Include. **N** – Ignore, do not include.
CURSORID	Displays a period, representing the encoded data that is your TIBCO Object Service Broker DB2 access statement. This data cannot be read or modified.

Cursor Names

@STATICSQL generates cursor names. Each cursor name contains the following information:

- The prefix D2
- The suffix specified when generating Static SQL (SGENP)
- A letter corresponding to the ACCESS field in the @SS_SELECTION table (N or G)
- The characters LS
- A number corresponding to the KEY field in the @SS_SELECTION table
- An identifier (N1 or N2)

Sample Warning Messages Generated By @STATICSQL

When you encounter warning messages in the message log similar to those shown in the example below, Static SQL is not generated for the lines named in the warning message, for example, LS9 and LS10. If you delete the entry from @SS_SELECTION, it reappears the next time you log your accesses for Static SQL. Therefore, you should set the INCLUDE flag for the corresponding numbered entry in the @SS_SELECTION table to N so that you do not see this warning message again, for example KEY=9 and KEY=10.

```
Unsupported selection: Table U40_DB2_EMP, cursor type "N", abbrev "LS9"
 UNSUPPORTED OPERATOR
 Unsupported selection: Table U40_DB2_EMP, cursor type "N", abbrev "LS10"
 UNSUPPORTED OPERATOR
 Generate/Remove Static SQL: 0 errors.  CHECK LOG for warnings
```

Sample Error Messages Generated By @STATICSQL

When you encounter error messages in the message log similar to the example below, no Static SQL is generated for the line named in the error message, for example, LS11. The first line displays the source of the problem (LS11). The second line of the message explains that the field accessed in the WHERE clause of the access logged (LS11) is not a field in the named TIBCO Object Service Broker DB2 table. Therefore, you should delete the corresponding numbered entry in the @SS_SELECTION table or set the INCLUDE flag for this entry to N.

```
Unsupported selection: Table U40_DB2_EMP, cursor type "N", abbrev "LS11"
 Field referenced in the selection is not part of table's defn
 Generate/Remove Static SQL: 0 errors.  CHECK LOG for warnings
```

When you generate Static SQL, the log tables @SS_SELECTION and @SS_ACCESSES are not cleared. When you modify an application, you must rerun only the modified portion of the application to collect the new accesses to TIBCO Object Service Broker DB2 tables. If you want to regenerate all accesses, you *must* clear the accesses logged in @SS_SELECTION for the TIBCO Object Service Broker DB2 table and delete the corresponding table name from @SS_ACCESSES. If you choose to remove the accesses, you must rerun all applications that access this table.

Task E: Assemble the Static SQL Handler

Using the Generated JCL

To assemble the Static SQL handler, use the JCL generated as member ASMDB2SS in the file specified in @DB2SERVERJCL when you run @STATICSQL.

The JCL keeps a history of all EXEC statements used to assemble the handler that were generated each time you ran @STATICSQL. Ensure that the proper EXEC statement (the one containing the *prefix/suffix* you chose when you ran @STATICSQL) is uncommented (MEMBER=*prefix/suffix*GEMP in the JCL mentioned above).

Task F: Link the Static SQL Handler

Sample JCL

To link the Static SQL handler, use the JCL generated as member LINKDB2S in the file specified in @DB2SERVERJCL when you run @STATICSQL.

The JCL keeps a history of all the EXEC statements used to link the handlers generated each time you ran @STATICSQL. Ensure that the proper EXEC statement (the one containing the *prefix/suffix* you chose when you ran @STATICSQL) is uncommented (MEMBER=*prefix/suffix*GEMP in the JCL mentioned above).

Default Static SQL Handlers

The generated Static SQL handler defaults to AMODE(31) and RMODE(ANY). If you use a Static SQL handler with the TIBCO Service Gateway for IMS/DB, you must add the default AMODE and RMODE to the link control card input statements as follows:

```
AMODE(31), RMODE(24)
```

The following JCL shows the changes to make to each link control card member generated by @STATICSQL if you are using the TIBCO Service Gateway for IMS/DB.

```
INCLUDE OBJECT2(D2CSGEMP)
MODE AMODE(31),RMODE(24)
ENTRY D2CSGEMP
NAME    D2CSGEMP(R)
```

Task G: Bind the DB2 Gateway Plan with Static SQL

Procedure

To bind the DB2 Gateway plan with Static SQL, use the sample JCL BINDDB2S included in the JCL library.

The XBINDDB6 member that is generated when you run @STATICSQL contains a list of the Static SQL handlers to be bound with the Gateway plan.

Task H: Update the Gateway Startup JCL

Add Static SQL Load Library to Startup JCL

To make the Static SQL handlers available to the Gateway, you must include the Static SQL load library in the //HDB2SSQL DD card of the Gateway startup JCL member DB2BATCH in the JCL data set. The Static SQL load library is not required in the STEPLIB and does not have to be authorized. The Gateway searches the STEPLIB if it does not find the Static SQL handler in //HDB2SSQL.

Static SQL Usage

Handling Static SQL Errors

Compile Errors

Compile errors encountered while assembling the Static SQL handler could be due to the following:

- If you use the extraction method to obtain the DB2 table definition (refer to Task A Choose a Method to Obtain the DB2 Table Definition on page 81) and change the actual DB2 table definitions without refreshing the TIBCO Object Service Broker tables containing the DB2 table definition information
- If you request an INSERT and a required DB2 field (NOT NULL) is not included in the TIBCO Object Service Broker DB2 table definition

Run Time Errors

Runtime errors encountered while accessing DB2 data could be due to the following:

- If you saved the TIBCO Object Service Broker DB2 table definition from the Table Definer and invalidated the Static SQL for this table. This occurs regardless of whether you made changes to the definition.

If no changes were made…	If changes were made…
Execute @STATICSQL (re-selecting the table) to reset the status for this table.	You must redo the procedure described in Chapter 3, Using Static SQL, on page 61.

- If the assembler code (member *prefix/suffix*) that contains the Static SQL statements generated when you ran @STATICSQL was manually modified.

Conditions Under Which Dynamic SQL is Used

The TIBCO Object Service Broker DB2 table definition must first qualify for Static SQL. If the Static SQL handler was generated, assembled, linked, and bound, and the TIBCO Object Service Broker table definition was not saved using the Table Definer since the handler was generated, Static SQL can be used.

If the TIBCO Object Service Broker DB2 table definition qualifies for Static SQL, the access you are attempting is compared to the corresponding Static SQL handler for the same selection. If the access is a GET or a FORALL with no selection criteria, a static sweep cursor is used. If the access (GET, FORALL, INSERT, DELETE, or REPLACE) you are attempting is found in the Static SQL handler, it is performed using Static SQL.

If the access is a GET or FORALL and is not found in the Static SQL handler, the selection criteria is examined. If a primary key equality or range cannot be extracted from the selection criteria so that it identifies a superset of the resulting set, Dynamic SQL is used. If it can be extracted and is found in the Static SQL handler, Static SQL is used (using this primary key equality or range as selection); otherwise, Dynamic SQL is used.

When accesses are logged, if a primary key range or equality can be extracted from the selection criteria, it is also logged.

Promoting Applications That Use Static SQL

To promote an application that uses Static SQL from your source system to a target system, complete the following steps:

1. Promote all TIBCO Object Service Broker DB2 tables in your application to the target system.
2. Extract the necessary occurrences of the TDS tables @SS_ACCESSES and @SS_SELECTION and promote them to the target system.
3. If the server ID on the target system is different than on the source system, modify the target server ID for each TIBCO Object Service Broker DB2 table in your application to match the source. For details, see Dynamically Changing Gateway Parameters on page 50.
4. If the DB2 Owner, DB2 Tablename, or DB2 Location on the target system are different than on the source system, modify the table @DB2TABLES accordingly.
5. Attach and start a Gateway that has the new server ID to the target system.
6. Use the @STATICSQL tool to regenerate the Static SQL for the required tables on the target TIBCO Object Service Broker system and the target DB2 subsystem.
7. Pre-compile, assemble, and link the generated static SQL assembler modules to the DB2 subsystem.

   This procedure is documented beginning in Task A: Log DB2 Accesses on page 63.

8. Bind the DB2 Gateway plan on the target DB2 subsystem.

   This procedure is documented in Binding the Gateway Plan on page 21.

You do *not* have to relog your TIBCO Object Service Broker DB2 access statements.

See Also *TIBCO Object Service Broker Managing Deployment* for more information on promoting objects.

Chapter 4 Managing DB2 Data Definitions

This chapter provides information to manage TIBCO Object Service Broker DB2 Data Definitions and to use the TIBCO Object Service Broker DB2 table.

Topics

This chapter describes how to perform tasks in TIBCO Object Service Broker in the text-based workbench. You can also perform development tasks through the TIBCO Object Service Broker UI, a graphical and text-based environment for TIBCO Object Service Broker development. For details on how to use the UI, see the TIBCO Object Service Broker UI online help.

Accessing DB2 Data from TIBCO Object Service Broker

To access DB2 data from TIBCO Object Service Broker, you must define a TIBCO Object Service Broker table of type DB2. A TIBCO Object Service Broker DB2 table can have one or more DB2 columns as fields, up to four DB2 columns as parameters, up to 16 DB2 columns as composite primary keys, and an optional location parameter.

Procedural Overview

The following list outlines the tasks required to define a TIBCO Object Service Broker DB2 table:

1. Task A Choose a Method to Obtain the DB2 Table Definition, page 81
2. Task B Invoke the Table Definer, page 82
3. Task C Select a DB2 Table or Stored Procedure, page 85
4. Task D Select DB2 Columns, page 91
5. Task E Change the Defaults if Necessary, page 95

These tasks are described in the following sections.

Task A Choose a Method to Obtain the DB2 Table Definition

There are two methods of obtaining DB2 table definitions. Refer to Obtaining DB2 Table Definitions on page 38 for more information. Your system administrator must choose one of the following two methods:

Gateway	Ensure that an instance of the Gateway is running before you define a TIBCO Object Service Broker DB2 table. This means that the DB2 table definition information is coming from SYSIBM.SYSTABLES/SYSIBM.SYSCOLUMNS for tables and from SYSIBM.SYSROUTINES/SYSIBM.SYSPARMS for stored procedures.
Extraction	Extract all DB2 table definition information and store it in TIBCO Object Service Broker.

Task B Invoke the Table Definer

Invocation Functions

Invoke the Table Definer from the workbench using the DT define table option or the primary command field. There are two functions you can perform:

- Access an existing definition
- Define a new TIBCO Object Service Broker DB2 table

These functions are described in the following sections.

Accessing Existing Tables

You can display the definition of an existing DB2 table from the workbench in one of three ways:

- 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 primary command field, for example:

  DT EMPLOYEE<Enter>
- Move the cursor to the DT define table option and press Enter. This displays the Object Manager screen, which contains a list of existing tables.

  Scroll through this list to see which table you require. To select a table, type **S** beside the name and press Enter.

If you are using the Gateway method to define TIBCO Object Service Broker DB2 tables (see Using the Extraction Method on page 40), you *must* have an instance of the Gateway running.

See Also The *TIBCO Object Service Broker Getting Started* manual for information on the Object Manager.

Defining a New Table

To define a new table, complete the following steps:

1. Type the name of a new TIBCO Object Service Broker DB2 table beside the DT define table option or in the primary command field and press Enter.

   A TDS definition template appears.

2. Change the Type field at the top of the screen to DB2 and press Enter.

   A screen similar to the one shown below appears.

```
COMMAND==>                          TABLE DEFINITION

        Table: EMPLOYEE_DB2     Type: DB2   Unit: VZO10

 Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: Y ImplUpd: N
 Subtype     : N (N/P/R)
 Creator     :
 Table       :
 DB2 Location:

   Location Parm    Typ Syn Len Dec Default          '      Event Rule   Typ Acc
   ---------------- --- --- --- -- ----------------  '   ---------------- -   -
 _                                                   '
                                                     ' _
                                                     '  _

 | ------------- DB2 Column --------------------------|------------ OSB Field ----
   Name                          Datatype Len Scale  |Num Name           Typ Syn
   ------------------------------ -------- ----- -Def|--- ---------------- - --



 (S=Select  P=Parameter  K=Key)
 PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

Screen Elements:

Header	The DB2 table or view on which to base your TIBCO Object Service Broker DB2 table. For details, see Task C Select a DB2 Table or Stored Procedure on page 85.
Location Parm	Allows you to specify a location parameter for the TIBCO Object Service Broker DB2 table. For details, see Task C Select a DB2 Table or Stored Procedure on page 85.

Event Rule	Allows you to specify an event rule for the TIBCO Object Service Broker DB2 table. For details, see Task C Select a DB2 Table or Stored Procedure on page 85.
DB2 Column and Metadata Field	Displays the columns of the DB2 table or the parameters of a procedure on which your TIBCO Object Service Broker DB2 table is based and your chosen TIBCO Object Service Broker fields. For details, see Task D Select DB2 Columns on page 91.
PF Keys	Displays the PF keys available from this screen. For details, see PF Keys on page 84.

PF Keys

The table below describes the commands (or abbreviations) that correspond to the PF keys:

PF Key	Primary Command	Description
-	**COPY**	Copies the definition of an existing TIBCO Object Service Broker DB2 table into the current definition.
PF1	-	Displays 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 DB2 Tables, on page 127 for more information.
PF3	**END**	Saves the existing definition and returns you to the workbench.
PF4	**DB2 TABLES**	Depending on the value of the Subtype field (N or P/R), lists the existing DB2 creators/tables or schemas/procedures, respectively, upon which you can base your TIBCO Object Service Broker DB2 table.
PF5	**SP_INFO**	Stored procedure information.
PF9	**SP_REFRESH**	Stored procedure result set creation.
PF12	**CANCEL**	Cancels changes to the definition and returns you to the workbench.
PF13	**PRINT**	Prints the definition of the table. You remain in the Table Definer.
PF22	**DELETE**	Deletes the definition of the TIBCO Object Service Broker DB2 table. You are prompted to confirm the deletion.

Task C Select a DB2 Table or Stored Procedure

Complete the necessary fields in the header, location parm, and event rule segments. These fields are described in the following sections.

Header Segment

The following fields are located in the header segment of the DB2 table definition screen.

Table	The table name specified when you invoked the Table Definer. Type a new name to save the definition of the current table under a new name. For more information on tools used to copy objects, refer to the *TIBCO Object Service Broker Shareable Tools* manual. Valid entries consist of 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 @ symbol denotes a table supplied by TIBCO Object Service Broker.
Type	The table type DB2, which you changed in Task B Invoke the Table Definer on page 82.
Unit	The user unit associated with the table. The unit marks a table as belonging to a particular application or to a logical unit such as utilities, accounting, or network control. The default unit for your user ID is specified in your TIBCO Object Service Broker user profile. Valid entries consist of a character string with a maximum length of 8 characters. These can be specified by your system administrator, for example, ACC.
Dictionary	Specifies whether to use the Internal (I) or External (X) dictionary.
Subtype	Identifies a Table (N), stored procedure (P) or result set (R) definition. Valid values: **N** or **P**. Note that result set definitions are generated using PF9.
Creator/ Schema	The name of the creator of the DB2 table. You can enter a search pattern in this field and then press PF4 to access a list of valid DB2 creators/schemas and their related tables/procedures. Refer to the **ServerID** field description for more information. **Note**: You can also set Dictionary=I in order to make use of the previously extracted DB2 metadata.
Table/ Procedure	Type the name of the DB2 table/procedure on which you want to base your TIBCO Object Service Broker DB2 table.

DB2 Location	The Gateway can execute only in a z/OS environment and connect to z/OS DB2; however, through z/OS, the Gateway can access remote DB2 subsystems connected to your local DB2 subsystem. Specify the name of the DB2 location where the DB2 data is stored, if this data is at a location other than your local DB2 subsystem.
ServerID	Type the ID for the Gateway or group of Gateways to use when accessing the table you are defining. This ID identifies a group of Gateways with common characteristics and must match the SERVERID startup parameter specified in the Gateway JCL. The default is DEFAULT. Refer to Gateway Parameters on page 42 for more information. The SERVERID parameter can be overridden at runtime. Refer to Dynamically Changing Gateway Parameters on page 50 for more information. Valid entries consist of a character string of up to eight characters.
Server Type	Type IM2 if you want to access both DB2 and IMS tables in a single transaction. The default specification is DB2.
Orders	Specifies whether the order clause is passed to DB2 in the select statement or ordering is done by TIBCO Object Service Broker. If the DB2 collating sequence is acceptable, specify Y. For release compatibility, the default is N. Type Y to pass the order clause to DB2 to perform the ordering before returning the data to TIBCO Object Service Broker. Type N for ordering to be performed in the Execution Environment. **Notes**: If you are using Static SQL, the static cursors available are searched for matching selection, as well as matching the sort clause. If a static cursor that satisfies both criteria is found, DB2 performs the ordering, otherwise TIBCO Object Service Broker does it. Refer to Chapter 3, Using Static SQL, on page 61 for more information. If using dynamic SQL, the Gateway generates dynamic SQL that includes the Ordered clause. Not used for stored procedure definitions.

Selecting the Foundation of a TIBCO Object Service Broker DB2 Table

To select a DB2 table as the foundation of your TIBCO Object Service Broker DB2 table, you can do one of the following:

- Type the name of a creator (schema) in the **Creator/Schema** field and a DB2 table (procedure) name in the **Table/Procedure** field, and press Enter.

- Type a search pattern (the wildcards * and ? are accepted; blank pattern is considered as *) in the **Creator** (Schema) and **Table** (Procedure) fields and press PF4 to access a list of all tables (procedures) pertaining to the creators (schemas) that fit your search patterns.

Sample Screen Using PF4

```
          List of available DB2 tables in e(X)ternal DB2 Dictionary
  Creator: SYS*
    Table: SQ*
--------------------------------------------------------------------------------
_ SYSIBM
  SQTCOLPRIVILEGES
_ SYSIBM
  SQTCOLUMNS
_ SYSIBM
  SQTFOREIGNKEYS
_ SYSIBM
  SQTPRIMARYKEYS
_ SYSIBM
  SQTPROCEDURECOLS
_ SYSIBM
  SQTPROCEDURES
_ SYSIBM
  SQTSPECIALCOLUMNS
_ SYSIBM
  SQTSTATISTICS
S=Select
 PFKEYS: 3 = SELECT & EXIT   ENTER = REFRESH   12 = EXIT
```

Type an **S** beside the desired table (procedure) and press Enter. A DB2 Table Definition screen similar to the following example displays. The column names for the selected DB2 table (procedure) are listed in the DB2 Column section of the screen.

```
COMMAND==>                    TABLE DEFINITION
Table       : CUSTOMER          Type: DB2   Unit: RFS
Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: Y ImplUpd: N
Subtype     : N (N/P/R)
Creator     : DSN8810
Table       : CUST
DB2 Location:

  Location Parm    Default         Src Sourcename   '   Event Rule      Typ Acc
  ---------------- --------------- --- ------------ '   ---------------- -   -
_ LOCATION                                          ' _
                                                    ' _

| ----------- DB2 Column --------------------------|-------- OSB Field -------
 Name                           Datatype Len Scale |Num Name           Typ Syn
 ------------------------------ -------- ----- -Def|--- ---------------- - --
_ CUST_ID                       CHAR         5   0 N
_ NAME                          CHAR        30   0 Y
_ ADDR_1                        CHAR        30   0 Y
_ ADDR_2                        CHAR        30   0 Y
_ CITY                          CHAR        15   0 Y
_ STATE                         CHAR         2   0 Y
_ OPEN_$                        DECIMAL      9   2 Y
_ PHONE                         CHAR        10   0 Y
(S=Select P=Parameter K=Key)
PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

DB2 Table Fields

Each column in the DB2 table or view contains information about it in the following fields:

Name	Name of the DB2 column.
DataType	Data type of the DB2 column.
Len	Length of the column as understood by DB2; for a decimal column, it is the precision (calculated in digits).

Scale	Number of digits to the right of the decimal for a decimal column, otherwise 0.
Def	If the column does not accept null values, whether or not DB2 provides a default value (NOT NULL WITH DEFAULT). Usually set to Y if the DB2 column does not accept null values and a default is supplied. A value of I or J indicates that this column is an IDENTITY column. In this case this field will not be included in INSERT and UPDATE statements. For stored procedures, this field has the following meaning: **P** – input value **O** – output value **B** – input/output value

Location Parameter Segment

You can use this section of the table definition screen to define a location parameter for a TIBCO Object Service Broker DB2 table. You use a location parameter to access DB2 data through a peer Gateway associated with another Data Object Broker (remote node). If you do not need to access remote data, use the **D** line command to delete the parameter. If you always access the DB2 table or view remotely, the node from which you request the access can have either a minimal or a full definition.

Event Rule Segment

Event rules enable you to validate data and automatically trigger other events based on specific update and/or retrieval access to TIBCO Object Service Broker DB2 tables. Event rules are always called when the table is accessed in the type of access specified. All rules applying to a specific access are executed in the order they appear in the scrollable event rule segment of the TIBCO Object Service Broker DB2 Table Definition screen. You define the fields in the event rule segment as follows:

Typ	Specifies the type of the event rule as follows: **V** – validation rule. Database updates are prohibited during the validation process. The rule must be a function that returns Y (yes), if the validation was successful, N (no), if the validation was not successful, or a message explaining why it was not successful. **T** – trigger rule. There are no restrictions on coding, other than the rule must not be a function, it cannot change the contents of the triggering row, and it cannot use the TRANSFERCALL statement. Nested triggers are possible.
Acc	The type of access, or manipulation, to be performed on the data, causing the event to be executed. Valid entries for validation rules: **W** – any write (insert, replace, delete); **I** – only insert; **R** – only replace; **D** – only delete. Valid entries for trigger rules: **W** – any write (insert, replace, delete); **I** – only insert; **R** – only replace; **D** – only delete; **G** – any retrieval.

See Also

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

TIBCO Object Service Broker Parameters for more information about parameters

Task D Select DB2 Columns

After selecting a DB2 table, you can select columns from it. On the Table Definition screen illustrated in Header Segment on page 85, place the cursor next to the DB2 column you want to select and use one of the following line commands:

P	Columns that are to be data parameters. You can choose up to four DB2 columns as data parameters. The Gateway forms a key out of these parameters (in sequence) followed by the TIBCO Object Service Broker primary key fields (in the order you specify in the **Num** field).
K	Columns that are to be the TIBCO Object Service Broker primary key fields. You can select up to 16 DB2 columns to form a composite primary key.
S	Columns to be fields in the TIBCO Object Service Broker DB2 table. If defining a stored procedure, all parameters are pre-selected as S. The number of fields you can select is dependent upon the length of the sum of all fields, primary keys, data parameters and control information. This sum must be less than or equal to 31 Kbytes. For an explanation of the formula used to calculate the total number of bytes of all fields, refer to the *TIBCO Object Service Broker Programming in Rules* manual. The number of fields you can access is dependent upon the CTABLESIZE Data Object Broker parameter. The ESTIMATETBLDFN rule, described in Estimating the CTABLESIZE Parameter on page 49, enables you to estimate the size of this parameter.

Select parameters and primary keys that, when combined, uniquely identify each DB2 occurrence, even though the DB2 table or view does not have unique keys defined for it. The Table Browser and Table Editor support up to eight composite primary keys. Use rules to access DB2 tables with more than eight primary keys.

TIBCO Object Service Broker Field Defaults

After selecting fields for the DB2 table, press Enter to display the default name, type, syntax, and length of these fields in the TIBCO Object Service Broker Field segment. When you press PF3 to save your selections and exit the Table Definer, a screen similar to the one shown here appears. The line commands remain visible, and you can change them by typing over them, or if you want to *ex*clude a selected DB2 column, you can de-select it by typing a blank over the line command.

```
COMMAND==>                    TABLE DEFINITION
Table        : CUSTOMER          Type: DB2   Unit: RFS
Dictionary   : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: Y ImplUpd: N
Subtype      : N (N/P/R)
Creator      : DSN8810
Table        : CUST
DB2 Location:

  Location Parm    Default         Src Sourcename   '   Event Rule      Typ Acc
  ---------------- --------------- --- ------------ '   ---------------- -   -
_ LOCATION                                          '  _
                                                    '  _

| ----------- DB2 Column ---------------------------|-------- OSB Field -------
 Name                           Datatype Len Scale  |Num Name            Typ Syn
 ------------------------------ -------- ----- -Def|--- ---------------- - --
K CUST_ID                       CHAR         5   0 N   1 CUST_ID            V
S NAME                          CHAR        30   0 Y   2 NAME               V
S ADDR_1                        CHAR        30   0 Y   3 ADDR_1             V
S ADDR_2                        CHAR        30   0 Y   4 ADDR_2             V
S CITY                          CHAR        15   0 Y   5 CITY               V
S STATE                         CHAR         2   0 Y   6 STATE              V
S OPEN_$                        DECIMAL      9   2 Y   7 OPEN_$             P
S PHONE                         CHAR        10   0 Y   8 PHONE              V
(S=Select P=Parameter K=Key)
PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

The screen above displays the default mapping of DB2 data types to TIBCO Object Service Broker semantic types and syntax. The default translations are shown in the following table. TIBCO Object Service Broker semantic types and syntax are described in *TIBCO Object Service Broker Programming in Rules*.

DB2 Column Type		TIBCO Object Service Broker Semantic Type	TIBCO Object Service Broker Syntax	TIBCO Object Service Broker Length
CHAR	CHAR		V	Same[a]
DATE	DATE	D	B	4
DECIMAL	DECIMAL		P	Note[c]
FLOAT	FLOAT	Not supported		
GRAPHIC	GRAPIC		UN	Note[d]
INTEGER	INTEGER		B	4
LONG VARCHAR	LONGVAR		V	Same
LONG VARGRAPHIC	LONGVARG		W	Note
SMALL INTEGER	SMALLINT		B	2
TIME	TIME		V	8[b]
TIMESTAMP	TIMESTMP		V	26[b]
VARCHAR	VARCHAR		V	Same
VARGRAPHIC	VARG		UN	Note

[a]If you select a character type DB2 column as a parameter, it defaults to a TIBCO Object Service Broker fixed-length character string since TIBCO Object Service Broker syntax V is not supported for TIBCO Object Service Broker parameters.

[b]DB2 passes all TIME and TIMESTAMP data to applications in the default character format defined for the DB2 subsystem. The format is a DB2 installation parameter.

[c]The DB2 DECIMAL type is defined with a precision specifying the total number of digits, and a scale specifying the number of decimal places. The TIBCO Object Service Broker length converts this to the appropriate number of packed bytes, and the number of decimal places remains the same. For example, a DB2 DECIMAL field of precision 8 and scale 2 is defined as length 5 and decimal places 2 in TIBCO Object Service Broker.

[d]DB2 graphic data types contain graphic symbols that consist of two bytes each. When you display these symbols in TIBCO Object Service Broker, they are mapped to TIBCO Object Service Broker character fields with twice that length.

Task E Change the Defaults if Necessary

You can modify any of the attributes in the **Metadata Field** section of the screen. These attributes describe the fields of your TIBCO Object Service Broker DB2 table. Each time you press Enter the screen is validated.

You can change the values of the following attributes:

- Default order in which the fields appear
- Default order of parameters

Default Order in Which the Fields Appear

This attribute is modified by typing alternate numbers in the **Num** field. The following applies by default:

- The primary key *must* be the first field. The DB2 columns that you choose for the primary key always appear first in the table, regardless of the numbers you type next to them in the **Num** field.
- Other selected fields follow the primary key in order from top to bottom of the list.
- If all selected fields are numbered and you choose another field and do not assign it a number, it is placed last in the order.

Default Order of Parameters

This attribute is modified by typing numbers in the **Num** field. The following applies by default:

- The parameters are ordered from top to bottom of the list.
- If all parameters are numbered and you choose another parameter without assigning it a number, it is placed last in the order.

Field Name

Type over the name in the **Name** field with a new name to uniquely identify the field within the TIBCO Object Service Broker DB2 table. You can use the same name as a field in another table. If you are moving data between this table and another table, giving fields the same names simplifies the process; however, we recommend that you use the same field name as the DB2 column name.

Valid entries: a character string (unique to the TIBCO Object Service Broker DB2 table definition) 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 (_).

TIBCO Object Service Broker Semantic Type and Syntax

Change the TIBCO Object Service Broker semantic type (**Typ** field) and syntax (**Syn** field) of the field. You can specify any valid semantic type and syntax combination that is supported for each DB2 data type. Valid combinations of type and syntax are described in *TIBCO Object Service Broker Programming in Rules*, and the TIBCO Object Service Broker syntax supported for each DB2 data type is listed in the following table.

Changing the default field syntax of a DB2 column can cause conversion overhead since the Gateway must convert each affected field of each row to the new syntax as defined in the TIBCO Object Service Broker DB2 table definition.

We recommend the following:

- If your DB2 Application Encoding Scheme is EBCDIC, define the DB2 character fields as TIBCO Object Service Broker syntax V.
- If your DB2 Application Encoding Scheme is Unicode, define the DB2 character fields as TIBCO Object Service Broker syntax UN. Conversion is performed from the single byte character set ID specified in your DB2 setup (the default being CCSID 1208, UTF-8) to UTF-16 in TIBCO Object Service Broker.
- Define as TIBCO Object Service Broker syntax RD the DB2 fields that have “FOR BIT DATA” set.

Data Conversions Supported

Only specific conversions from DB2 data types to TIBCO Object Service Broker syntax are supported by the Gateway as shown in this table.

TIBCO Object Service Broker Syntax --> DB2 Data Type	Fixed-Length Character String	Packed Decimal	Binary	Variable-Length Character String	Unicode	Raw Data
CHARACTER	Y			Y	Y	
DATE	Y		Y			

TIBCO Object Service Broker Syntax --> DB2 Data Type	Fixed-Length Character String	Packed Decimal	Binary	Variable-Length Character String	Unicode	Raw Data
DECIMAL		Y	Y			
FLOAT						
GRAPHIC	Y			Y		Y
INTEGER		Y	Y			
LONG VARCHAR	Y			Y	Y	
LONG VARGRAPHIC	Y			Y		Y
SMALL INTEGER		Y	Y			
TIME	Y					
TIMESTAMP	Y					
VARCHAR	Y			Y	Y	
VARGRAPHIC	Y			Y		Y

Additional Considerations

- Conversions to and from TIBCO Object Service Broker floating point syntax are not supported.
- If DB2 data contains lowercase characters, define the corresponding TIBCO Object Service Broker field with case-sensitive syntax V.
- If a selected DB2 column does not support nulls, it is a required field. For DB2 character fields defined with TIBCO Object Service Broker syntax C, trailing blanks become nulls. Therefore, define the TIBCO Object Service Broker syntax as V to send a field containing only blanks to the Gateway.

Field Length

Change the length in the **Len** field and, if applicable, the number of digits to the right of the decimal in the **Dec** field. The data is padded or truncated as necessary.

Occurrence Order

Specify A (ascending) or D (descending) in the **Ord** field to determine the order of the occurrences.

Server Orders

If Server Orders=Y, DB2 passes data to TIBCO Object Service Broker already ordered and TIBCO Object Service Broker does not re-order the data.

If Server Orders=N, the following occurs:

- For a qualified GET or FORALL, all DB2 data that matches the selection criteria must be retrieved and sorted by the Execution Environment.
- For an unqualified GET or FORALL, all DB2 data must be retrieved and sorted by the Execution Environment before data is passed back to a rule or a TIBCO Object Service Broker tool.

Sample External Table Definition to Access DB2 Data

After modifying the default TIBCO Object Service Broker field attributes, a screen similar to the following displays:

```
COMMAND==>                   TABLE DEFINITION
Table       : CUSTOMER          Type: DB2   Unit: RFS
Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: Y ImplUpd: N
Subtype     : N (N/P/R)
Creator     : DSN8810
Table       : CUST
DB2 Location:

  Location Parm    Default         Src Sourcename    '   Event Rule      Typ Acc
  ---------------- --------------- --- ------------ '   ---------------- -   -
_ LOCATION                                          ' _
                                                    ' _

| ----------- DB2 Column ----------|-------- OSB Field ---------------------------
 Name            Datatype Len Scale |Num Name    Typ Syn Len  Dec Req Default
 -------------- -------- ----- -Def|--- --------- - --  ----- --Ord- -------------
K CUST_ID        CHAR         5   0 N   1 CUST_ID     V       5
S NAME           CHAR        30   0 Y   2 NAME        V      30    A
S ADDR_1         CHAR        30   0 Y   3 ADDR_1      V      30
S ADDR_2         CHAR        30   0 Y   4 ADDR_2      V      30
S CITY           CHAR        15   0 Y   5 CITY        V      15
S STATE          CHAR         2   0 Y   6 STATE       V       2
S OPEN_$         DECIMAL      9   2 Y   7 OPEN_$      P       4  2
S PHONE          CHAR        10   0 Y   8 PHONE       V      10
(S=Select P=Parameter K=Key)
PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

If you are satisfied with your DB2 table definition, press PF3 to save the definition. If you do not want to save your changes, press PF12 to cancel them.

DB2 Stored Procedure Result Set Extract

Press PF9 = RS_REFRESH to generate TIBCO Object Service Broker table definitions mapping the result sets, if any, produced by a stored procedure. The following screen is displayed:

```
Result Set Mapping Construction for Procedure mapped by P

Enter valid input to run DB2 Stored Procedure EMPRSETC
Warning: This process executes the Stored Procedure.
         Be aware that your Stored Procedure may update data.


Argument Name      Value
PDEPTNAME
PSQLCODE
PSQLSTATE
PSQLERRMC


PFKEYS: ENTER=EXTRACT 3=SAVE & RETURN 12=CANCEL & RETURN
```

To describe the result sets of a stored procedure, the stored procedure must be executed. If the stored procedure has input values, you must provide a set of input values that make the procedure return the result set(s). Note that by executing the stored procedure, you may be updating data.

If prompted for input values, specify them and press Enter. You will then be prompted to confirm the execution of the stored procedure for the specified set of input values. Upon confirmation (PF22), the procedure is executed and you will see displayed the values it has returned in its output and input/output parameters. In addition, a screen message will be displayed reporting the number of result sets built.

This process may be repeated any number of times. The table definer stores in a temporary repository the descriptions of the result sets built by this process. You may press PF12 = CANCEL to clean up this repository and return to the Table Define Screen, or PF3 = SAVE & EXIT to generate the persistent OSB table definitions mapping those result sets.

Result Set Table Naming Convention

Under DB2 conventions, any result set returned by a stored procedure has a unique (within the scope of the stored procedure) name (up to 30 bytes long) assigned by the stored procedure. The TIBCO Object Service Broker table definer uses these names and the TIBCO Object Service Broker name of the procedure mapping being defined when generating result set mappings.

Result set tables are named using the following considerations:

- If the concatenation of the TIBCO Object Service Broker name and the result set name does not exceed 16 characters, it is used to name the result set mapping.
- Otherwise, a unique table name DB2_RSET_<n> is generated.

For example, if the TIBCO Object Service Broker table PROC maps the DB2 stored procedure EMPRSET that returns two result sets named CSR1 and CSR2, the names of TIBCO Object Service Broker tables generated by the definer will be PROCCSR1 and PROCCSR2.

Stored Procedure Definition

In the Subtype field, specify the value **P**. Enter a schema and procedure name, or use the PF4 key to see a list of stored procedures.

The columns displayed are the parameters of the stored procedure. Two additional TIBCO Object Service Broker fields, @HANDLE@ and #RS#, are added to the definition. All fields are preselected; do not deselect any fields, as this will cause the call to the procedure to fail.

If the stored procedure has result sets available to the calling program, use the SP_INFO and SP_REFRESH PF keys to list and to automatically create table definitions for them.

```
COMMAND==>                          TABLE DEFINITION

       Table: PROC            Type: DB2   Unit: VZO10

Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: N ImplUpd: N
Subtype     : P (N/P/R)  #Parms: 5       #ResultSets: 1      CommitOnReturn: N
Schema      : DEVL7083
Procedure   : EMPRSETC
DB2 Location:

  Location Parm    Default         Src Sourcename    '      Event Rule   Typ Acc
  ---------------- ---------------- - ------------- '   ---------------- -   -
_ LOCATION                                          ' _
                                                    ' _

| ----------- DB2 Column ----------------------------|-------- OSB Field -------
  Name                           Datatype Len Scale |Num Name            Typ Syn
  ------------------------------ -------- ----- -Def|--- ---------------- - --
K @HANDLE@                       SMALLINT     2  0      1 @HANDLE@           B
K #RS#                           SMALLINT     2  0      2 #RS#               B
K PDEPTNO                        CHAR         3  0 P    3 PDEPTNO            V
S PDEPTNAME                      VARCHAR     36  0 O    4 PDEPTNAME          V
S PSQLCODE                       INTEGER      4  0 O    5 PSQLCODE           B
S PSQLSTATE                      CHAR         5  0 O    6 PSQLSTATE          V
S PSQLERRMC                      VARCHAR    250  0 O    7 PSQLERRMC          V


(S=Select  P=Parameter  K=Key),
PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

Effects of DB2 Column Selection on DB2 Processing

Take the following effects into consideration when selecting DB2 columns for a TIBCO Object Service Broker DB2 table definition:

- Only the DB2 columns you select in your TIBCO Object Service Broker DB2 table definition are requested from DB2. Consequently, if these columns reside in the DB2 index, DB2 could return results without having to read its data pages.
- If you need to insert rows to DB2 tables, ensure your DB2 table definition includes all NOT NULL (without default) fields.
- If you choose DB2 columns as parameters that correspond to a unique key of a DB2 table, you can insert *only* a single row using this definition. This is because the Gateway combines the parameter values with the TIBCO Object Service Broker primary key values, causing an attempt to insert a duplicate row to DB2.
- The maximum occurrence length of the defined expanded fields must be less than or equal to 31744 bytes.

Chapter 5 Processing Data

This chapter provides information to access the TIBCO Object Service Broker DB2 tables.

Topics

Accessing TIBCO Object Service Broker DB2 Tables

When you access DB2 data from TIBCO Object Service Broker, the following occurs:

- TIBCO Object Service Broker requests are translated into SQL statements. The Gateway generates Dynamic SQL to access DB2 data, unless Static SQL is in place for a TIBCO Object Service Broker DB2 table. Existing Static SQL is used by the Table Browser, Table Editor, or a new application. Refer to Chapter 3, Using Static SQL, on page 61 for more information.
- DB2 column data types are translated to the TIBCO Object Service Broker field types defined in the TIBCO Object Service Broker DB2 table.

You can access the data using:

- Table Browser and Table Editor
- Rules language

These methods are described in the following sections.

Using the TIBCO Object Service Broker Table Browser and Table Editor

You can use the Table Browser to browse a defined TIBCO Object Service Broker DB2 table by typing the table name next to the BR browse table option, for example:

```
EMPLOYEE(1998)<Enter>
```

You cannot use the table browser or table editor to view stored procedure tables.

Sample Screen

In the sample screen below, EMPLOYEE is the name of the TIBCO Object Service Broker DB2 table, and 1998 is the parameter value for the table instance. The DB2 data is presented in the TIBCO Object Service Broker table format:

```
BROWSING TABLE   :  EMPLOYEE(1998)
COMMAND ==>
                                                                  SCROLL: P
  EMPNO          LNAME               POSITION      MGR#  DEPTNO  SALARY
_ ------ -------------------- -------------- ------ ------ ---------
_  22001 DRABEK               CUST SUPPORT    56112     30    900.00
_  22007 ROEDER               CUST SUPPORT    56112     30    900.00
_  30058 HOEGSON              PRE-SALES       37219     20    675.00
_  34111 TERAMURA             PRE-SALES       37219     20    710.00
_  34121 LEES                 CUST SUPPORT    56112     30    700.00
_  36162 MORANG               JR OPERATOR     44798     80    575.00
_  41001 CROFTON              TECH WRITER     80002     70    675.00
_  41007 STEVENSON            EDUCATOR        80002     60    700.00
_  41009 SMITH                TESTER          79912     50    600.00
_  44385 SOUZA                SALES           37219     10    719.00
_  44622 SAUNDERS             ACCOUNTANT      98895     40    800.00
_  51111 HRODEK               ANALYST         79912     50    710.00
_  51121 CANNON               ANALYST         79912     50    700.00
_  51162 KIMURA               JR PROGRAMMER   79912     50    575.00
_  61219 WONG                 SENIOR ANALYST  79912     50    820.00
_  61385 DHILLON              EDUCATOR        80002     60    685.00
_  61622 SCHULTZ              SENIOR ANALYST  79912     50    800.00
PFKEYS: 1=HELP 5=FIND NEXT 9=RECALL 18=EXCLUDE 13=PRINT 3=END 14=EXPAND
At TOP
```

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

Exceptions

You can browse and edit a TIBCO Object Service Broker DB2 table in the same way you would browse or edit another TIBCO Object Service Broker table except in the following cases:

- If a selected column has an N in the **Upd** field, you cannot edit this field. You can, however, edit other fields in the same table.
- Depending on its indexes, you can use DB2 to insert duplicate primary keys. This results in the following:
  - If you delete an occurrence using the Table Editor, *the first row* in the DB2 table to match the parameters (if any) and primary key values is deleted.
  - If you update an occurrence, *all rows* that match the parameters (if any) and the primary key values in the DB2 table are updated.
- If your table definition contains fields of syntax C or V that are longer than 260 bytes, the following restrictions apply:
  - You must use the Single Occurrence Editor from the Table Editor to edit them.
  - You must use **`SELECT LIKE`** instead of **`SELECT`** to access fields of this length.
- Using the Single Occurrence Editor from the Table Browser begins a dependent transaction in TIBCO Object Service Broker, assigning a new Gateway. Therefore, if you edit an occurrence using the Single Occurrence Editor from the Table Browser it creates a locking conflict in DB2 since the Table Browser transaction already has a shared lock on the DB2 data. You *can* use the Single Occurrence Editor from the Table Editor, since this does not begin a dependent TIBCO Object Service Broker transaction and requires only one Gateway.

  The Table Browser does *not* hold locks on TDS data; however, this is not true for TIBCO Object Service Broker DB2 tables. Locking of DB2 data is determined by the Gateway BIND parameters and the DB2 subsystem, not by TIBCO Object Service Broker.

Using Rules

Accessing DB2 data using the TIBCO Object Service Broker rules language is similar to accessing TIBCO Object Service Broker data. The main difference is in the way DB2 interprets the request. Use the following sections in conjunction with *TIBCO Object Service Broker Programming in Rules*, which describes rules statements and coding.

Transaction Processing

If you issue a TIBCO Object Service Broker EXECUTE statement within a main (parent) transaction, it creates another transaction stream (child transaction), to a maximum of ten streams. The number of streams allowed in a TIBCO Object Service Broker transaction depends on the Execution Environment parameter TRANMAXNUM, which has a default of nine streams. Each transaction stream in TIBCO Object Service Broker accessing DB2 data requires its own Gateway thread.

Ensure your system administrator is aware of the number of Gateway threads required to accommodate all transaction streams accessing DB2 data in a single transaction.

Using TRANSFERCALL or DISPLAY & TRANSFERCALL statements in a rule minimizes Gateway threads and reduces the possibility of DB2 locking contention.

Transaction Limitations

The number of TIBCO Object Service Broker DB2 tables you can access per transaction depends on the POOLSIZE gateway parameter. For details, see Gateway Parameters on page 42.

If you use the default parameter values, you can access at least 16 TIBCO Object Service Broker DB2 tables per transaction; possibly more, depending on the size of the TIBCO Object Service Broker DB2 table definitions. Refer to Estimating the CTABLESIZE Parameter on page 49 for more information.

Rules Behavior

The following sections outline differences encountered while using rules and also point out normal TIBCO Object Service Broker rules behavior that you must consider when building applications.

Retrieval Processing

A single SQL cursor is used to retrieve data from DB2 for each retrieval statement (GET or FORALL) in your rule.

If your DB2 subsystem is version 2.2 or later, and you are using Static SQL to access DB2 data from TIBCO Object Service Broker, FOR FETCH ONLY is included in each SELECT cursor.

When a TIBCO Object Service Broker transaction runs in browse mode, locks are *not* taken on the TIBCO Object Service Broker data; however, locks *are* taken on the DB2 data in accordance with how the Gateway plan is bound and how the DB2 table was created.

GET Statement

A GET statement retrieves the first occurrence in the TIBCO Object Service Broker DB2 table that satisfies the specified selection criteria.

- If you are using Static SQL to access DB2 data from TIBCO Object Service Broker, OPTIMIZE FOR 1 ROW is included in each SELECT cursor associated with a GET statement.

  A GET...ORDERED statement must retrieve all DB2 data that satisfies the selection criteria and SORT it in the Execution Environment before returning the first occurrence that meets the selection criteria. OPTIMIZE FOR 1 ROW is not included in the SELECT cursor associated with a GET...ORDERED statement.

- If your table definition specified DB2 Ordering (Orders=Y), a GET …ORDERED returns only the occurrence that meets the selection criteria from the sorted output, instead of returning all the occurrences to TIBCO Object Service Broker for sorting and selection.

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 names.

Rows are returned to TIBCO Object Service Broker in the order in which DB2 passes them. If you require a different order, you must include an ORDERED clause in your FORALL statement. TIBCO Object Service Broker orders only rows specified in the selection criteria.

For Dynamic SQL, a pool of five cursors is shared for all requests for a sequence of rows (TIBCO Object Service Broker FORALL statements) in a single transaction. Therefore, a transaction can have up to five nested FORALL statements where the resulting set of occurrences from each FORALL is greater than 4 KB. If you are using Static SQL, you can have more than five nested FORALL statements.

- If your table definition specified DB2 Ordering (Orders=Y), a FORALL …ORDERED means that DB2 can use its own sorting mechanisms to maximize efficiencies of the DB2 database and indexes. Only the rows that satisfy the Ordered request are returned to TIBCO Object Service Broker instead of all the data.
- If you are using Static SQL, the gateway converts Unicode data to EBCDIC from host variables, even for fields defined as syntax UN. If the Unicode data cannot be converted, the FORALL statement fails with a conversion error.

Replace (Update) Processing

The occurrence that matches the selection criteria (based on the primary key and parameters) is updated if you specify a unique DB2 occurrence. If you do *not* specify a unique DB2 occurrence, all DB2 rows that match this criteria are updated.

- Select parameters and TIBCO Object Service Broker primary keys that, when combined, enable each DB2 occurrence to be uniquely identified, even though the DB2 table does not have unique keys defined. Do not include non-updateable DB2 columns (N in the **Upd** field) as fields in a TIBCO Object Service Broker DB2 table if you plan to update DB2 data using this definition.
- If you are using Static SQL, the gateway converts Unicode data to EBCDIC to populate host variables, even for fields defined as syntax UN. If the Unicode data cannot be converted, the REPLACE statement fails with a conversion error.

Insert Processing

If you need to insert rows to a DB2 table or view, ensure your DB2 table definition includes all NOT NULL (without default) fields and that these fields are initialized before the INSERT statement.

If you choose DB2 columns as parameters that correspond to a unique key of a DB2 table or view, you can insert only a single row using this definition. This is because the Gateway combines the parameter values with the TIBCO Object Service Broker primary key values, causing an attempt to insert a duplicate row to DB2.

If you are using Static SQL, the gateway converts Unicode data to EBCDIC to populate host variables, even for fields defined as syntax UN. If the Unicode data cannot be converted, the INSERT statement fails with a conversion error.

Delete Processing

If you delete an occurrence using the Table Editor, the first row in the DB2 table or view to match the parameters (if any) and the primary key values is deleted.

If you edit the occurrence using the Single Occurrence Editor, the first row in the DB2 table or view to match the parameters (if any) and the primary key values appears for editing.

To clear all the data in a DB2 table, use the $CLRTAB shareable tool.

See Also *TIBCO Object Service Broker Shareable Tools* for information on the $CLRTAB tool.

Use of LIKE

If your DB2 gateway startup parameter specifies USEDB2LIKE, a FORALL…LIKE means that DB2 LIKE statements are generated to maximize efficiencies in retrieval. Only rows that satisfy DB2 LIKE evaluation are returned. Because of differences in evaluation processing, use of DB2 LIKE could return a different number of rows than use of a TIBCO Object Service Broker LIKE. Also, you can use LIKE on all TIBCO Object Service Broker field types. DB2 accepts only LIKE on CHAR, VARCHAR, and graphic types. For more information refer to Using DB2 LIKE or NOT on page 65.

DB2 LIKE Statement Format

To use the DB2 LIKE statement format (not to be confused with the TIBCO Object Service Broker LIKE) you must apply the following two settings:

- For dynamic SQL, to make the DB2 Gateway formulate DB2 LIKE clauses you must specify USEDB2LIKE in the DB2 gateway start up parameters. Without this specification, the Gateway parses the rows returned.
- When generating Static SQL using the @STATICSQL tool, to include LIKE statements in the Static handler, you must sign on to the Execution Environment using USEDB2LIKE(Y). LIKE statements are generated in the

static SQL handler. Without this specification, LIKE statements are not included in the static handlers.

WHERE Clause Processing and NULLS

The Gateway translates the following on WHERE clauses referring to NULL fields:

TIBCO Object Service Broker Operand	Translates to SQL...
= NULL	IS NULL
¬= NULL	IS NOT NULL
> NULL	IS NOT NULL
<= NULL	IS NULL

All other operands are discarded.

WHERE clauses containing the operand >= NULL are disregarded because they imply selection on the entire table.

WHERE Clause Processing and Parameterized Tables

Depending on how a WHERE clause for a parameterized table is coded, the Gateway can generate different SQL for parameterized table accesses. The results of the SQL processing are the same.

The following examples show the differences in the generated SQL:

```
FORALL TABLE WHERE PARAM='value' AND FIELD1='value'
```

generates the SQL statement:

```
SELECT TABLE WHERE PARAMFIELD='value'
```

Whereas the following, which is the recommended syntax to use:

```
FORALL TABLE('value')WHERE FIELD1='value'
```

generates the SQL statement:

```
SELECT TABLE WHERE PARAMFIELD='value' AND FIELD='value'
```

Invoking DB2 Stored Procedures

This section describes how to invoke DB2 stored procedures.

Overview

A DB2 stored procedure is an executable unit that DB2 callers can have DB2 invoke on their behalf. A DB2 stored procedure as seen by the caller has the following characteristics:

- Schema (up to 128 bytes long).
- Name (up to 128 bytes long).
- Number of positional parameters that have the following attributes:
  - Parameter kind (**P** – IN value, **O** – OUT value, **B** – INOUT value).
  - Name (up to 30 bytes long).
  - DB2 datatype.
- Number of result sets returned (a result set is a DB2 cursor available for fetching upon execution of the stored procedure).
- The Commit on Return attribute indicating if DB2 will COMMIT prior to returning control upon execution of the stored procedure.

The TIBCO Object Service Broker implementation of DB2 stored procedures support allows the following:

- The TIBCO Object Service Broker table definer to define a TIBCO Object Service Broker table of type DB2, subtype P mapping the signature of a DB2 stored procedure (uniquely identifying the stored procedure and its parameters to be used for invoking it via DB2).
- The TIBCO Object Service Broker table definer to define TIBCO Object Service Broker tables of type DB2, subtype R mapping the result sets produced by a DB2 stored procedure.
- The rules language to execute a DB2 stored procedure and access its OUT parameters and INOUT parameters.
- The rules language to fetch the result sets produced by a DB2 stored procedure.

TIBCO Object Service Broker table definitions of type DB2, subtype P, are used to map DB2 stored procedures. When such a definition is created by the TIBCO Object Service Broker table definer, the stored procedure being mapped is fully identified by the Schema/Procedure pair and, if the Service Gateway for DB2 denoted by SERVERID is running and available, the following attributes are fetched from DB2 by the definer:

- #Parms (informational) – number of procedure parameters as stored in DB2's table SYSROUTINES; includes the IN, OUT and INOUT parameters.
- #ResultSets (informational) – number of result sets to be returned by the stored procedure as stored in DB2's table SYSROUTINES.
- CommitOnReturn – the logical value stored in DB2's table SYSROUTINES.
- The stored procedure's parameters described in DB2's SYSPARMS table; they are treated similarly to DB2 columns in the case when the definer is used to map regular DB2 tables. The definer enforces fields @HANDLE@ and #RS# as the first two fields of the definition and marks them with the K line command (as primary keys).

TIBCO Object Service Broker table definitions of type DB2, subtype R, are used to map DB2 stored procedures' result sets. Such definitions are generated by the definer (the RS_REFRESH PF key) when defining a stored procedure mapping; those pertaining to a particular DB2 stored procedure can be viewed via the RS_INFO key, which displays a list of TIBCO Object Service Broker names for result set mapping, their respective DB2 stored procedure name, and TIBCO Object Service Broker mapping of the latter. If an entry of this list is selected with the S line command, the field structure of the result set is displayed in Browse mode. In order to modify a TIBCO Object Service Broker table definition of type DB2, subtype R, use the table definer directly against it.

The ImplUpd (implied update) attribute indicates whether TIBCO Object Service Broker is expected to consider the fact of invocation of this stored procedure as an update operation, meaning that the transaction in whose context it occurred should be reported to the Data Object Broker as resulting in data changes, which is essential for determining the specifics of commit/rollback/recovery procedures to be used for this transaction. Note that ImplUpd = N improves the application's performance (unless other modifications of data, either implicit or explicit, occur within the boundaries of the same transaction).

Language Provisions

In the TIBCO Object Service Broker implementation of DB2 stored procedures support, TIBCO Object Service Broker table definitions of type DB2, subtype P, differ from all other tables of type DB2. For tables of type DB2, subtype P, all tabular access verbs of the rules language (GET, FORALL, INSERT, DELETE, REPLACE) are considered illegal operations, whereas the CALL statement accepts the name of such a table as an executable entity.

Run a DB2 stored procedure with the following `CALL` statement in either positional or keyword form:

```
CALL <proc> WHERE P1=<p1> & P2=<p2> & ... & PN = <pn>
```

where <proc> is the name of a TIBCO Object Service Broker table of type DB2, subtype P; P1, P2, ..., PN are the names of fields in that table; and <p1>,<p2>,…,<pn> are values/expressions to be passed as IN and INOUT parameters of the procedure. The OUT parameters must be omitted from the list; if no IN and INOUT parameters are defined, the parentheses (or the WHERE clause) must also be omitted.

If the keyword notation is used and NULL is to be passed as the value for a parameter, say PI, you can omit the respective term (& PI = NULL).

Note: Although NULL and zero-length string are considered identical in the context of TIBCO Object Service Broker rules, the DB2 gateway differentiates between them when passing parameter values to DB2-stored procedures. Thus, CALL PROC(NULL) is identical to CALL PROC WHERE P1=NULL and to CALL PROC; the DB2-stored procedure <proc> receives NULL as input value. However, CALL PROC('') or CALL PROC WHERE P1='' must be issued to have a zero-length string passed to the stored procedure.

Upon return, the values set up by the stored procedure are available to the caller via the regular <table>.<field> notation. If any result sets have been generated, the caller must know the names of the TIBCO Object Service Broker tables (tables of type DB2, subtype R) for mapping them and use the FORALL statements to fetch their contents.

Two auxiliary fields, @HANDLE@ and #RS#, are always included (and marked K – key) in the definition of a TIBCO Object Service Broker stored procedure mapping. They have the following meaning:

- @HANDLE@ is an internally-assigned value, available upon return from the procedure, that uniquely identifies the procedure invocation context created for this particular call. This context is kept for subsequent result set fetches until explicitly discarded or implicitly cleaned up at transaction end.
- #RS# is the actual number of result sets created by the stored procedure during this particular call. If #RS# is set to 0 by the CALL statement, the procedure call context has already been discarded automatically.

The @HANDLE@ auxiliary field is always included (and marked P – data parameter) in the definition of a TIBCO Object Service Broker result set mapping. At the result set fetch time, the caller is expected to supply as input for @HANDLE@ the value assigned to the @HANDLE@ field by the CALL statement.

Example

This example calls a stored procedure and fetches the result set it produces. EMPRSET is a DB2-provided stored procedure sample. The TIBCO Object Service Broker table definer displays it as shown below.

Mapping of PROC:

```
COMMAND==>                          TABLE DEFINITION

        Table: PROC             Type: DB2   Unit: VZO10

 Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: N ImplUpd: N
 Subtype     : P (N/P/R)  #Parms: 5       #ResultSets: 1      CommitOnReturn: N
 Schema      : DEVL7083
 Procedure   : EMPRSETC
 DB2 Location:

   Location Parm    Default        Src Sourcename     '      Event Rule   Typ Ac
   ---------------- ---------------- - ------------- '   ---------------- -   -
 _ LOCATION                                          ' _
                                                     ' _

 | ----------- DB2 Column ----------------------------|-------- OSB Field ------
   Name                           Datatype Len Scale |Num Name            Typ Sy
   ------------------------------ -------- ----- -Def|--- ---------------- - --
 K @HANDLE@                       SMALLINT     2  0      1 @HANDLE@           B
 K #RS#                           SMALLINT     2  0      2 #RS#               B
 S PDEPTNO                        CHAR         3  0 P    3 PDEPTNO            V
 S PDEPTNAME                      VARCHAR     36  0 O    4 PDEPTNAME          V
 S PSQLCODE                       INTEGER      4  0 O    5 PSQLCODE           B
 S PSQLSTATE                      CHAR         5  0 O    6 PSQLSTATE          V
 S PSQLERRMC                      VARCHAR    250  0 O    7 PSQLERRMC          V

 (S=Select  P=Parameter  K=Key)
 PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

The mapping for the result set, generated under the name PROCC1, is shown below.

Mapping of PROCC1:

```
COMMAND==>                          TABLE DEFINITION

        Table: PROCC1           Type: DB2   Unit: DB2_SP

 Dictionary  : X (X/I)  ServerID: DEFAULT  ServerType: DB2 Orders: N ImplUpd: N
 Subtype     : R (N/P/R)
 Schema      : DEVL7083
 Procedure   : EMPRSETC
 DB2 Location:                      CursorName: C1

   Location Parm    Default         Src Sourcename     '      Event Rule   Typ Acc
   ---------------- ----------------- - ------------- '   ---------------- -   -
 _ HHH                                                ' _
                                                      ' _

 | ----------- DB2 Column ---------------------------|-------- OSB Field -------
   Name                           Datatype Len Scale |Num Name            Typ Syn
   ------------------------------ -------- ----- -Def|--- ---------------- - --
 P @HANDLE@                       SMALLINT     2   0     1 @HANDLE@           B
 S EMPNO                          CHAR         6   0     1 EMPNO              V
 S FIRSTNME                       VARCHAR     12   0     2 FIRSTNME           V
 S MIDINIT                        CHAR         1   0     3 MIDINIT            V
 S LASTNAME                       VARCHAR     15   0     4 LASTNAME           V
 S HIREDATE                       DATE        10   0     5 HIREDATE         D B
 S SALARY                         DECIMAL      9   2     6 SALARY             P


 (S=Select  P=Parameter  K=Key)
 PFKEYS: 4=DB2 TBLS 3=END 12=CANCEL 2=DOC 22=DELETE 5=RS_INFO 9=RS_REFRESH
```

A rule that executes this procedure and fetches its result set could be similar to the following:

```
_ LOCAL T;
_ ------------------------------------------------------
_ ------------------------------------------------------+---
_ CALL PROC('D11');                                     | 1
_ FORALL PROCC1(PROC.@HANDLE@) :                        | 2
_    T = '';                                            |
_    FORALL $$SYSFIELDS('PROCC1') :                     |
_       T = T || PROCC1.($$SYSFIELDS.NAME) || ' ';      |
_       END;                                            |
_    CALL MSGLOG(T);                                    |
_    END;                                               |
_ ------------------------------------------------------|
```

Note that after the CALL statement, the following values have been assigned:

```
PROC.@HANDLE@ = <non-NULL>
PROC.#RS#     = 1

PROC.PDEPTNAME = MANUFACTURING SYSTEMS
PROC.PSQLCODE  = 0
PROC.PSQLSTATE = 00000
PROC.PSQLERRMC = <NULL>
```

Querying the Layout of Result Sets

The caller can dynamically query the layout of the result set(s) produced by a stored procedure. To do so, after the `CALL PROC(…)` statement, issue the following:

```
FORALL @SYSRSCOLS(PROC.@HANDLE@,n),
```

where 0 < n <= PROC.#RS# is the number of the result set being queried. The sequence (numbering) of result sets is determined by the procedure. The table @SYSRSCOLS is a subview of @SYSRSCOLUMNS and comprises all columns of a particular result set (identified by the procedure-assigned name CURSOR_NAME) with their attributes as defined by the stored procedure.

Discarding Stored Procedure Context

When the context of a stored procedure call is no longer needed, you can release the resources used by the context, while the transaction proceeds. After the `CALL PROC(…)` statement, issue the following:

```
CALL @DB2PROCDISCARD(PROC.@HANDLE@).
```

Behavior of the Gateway

TIBCO Object Service Broker DB2 Requests

This section describes how the Gateway handles TIBCO Object Service Broker requests with respect to:

- Synchronization and recovery
- Static SQL
- Error handling

Synchronization and Recovery

Locking DB2 Data

The Gateway BIND parameters and the DB2 subsystem, not TIBCO Object Service Broker, determine how DB2 data is locked. A DB2 transaction spans the same length of time as a TIBCO Object Service Broker transaction.

When a commit is issued, DB2 data is committed and locks are released. The cursor position is maintained, because the DB2 statement is coded using WITH HOLD (as in `EXEC SQL DECLARE HURC01 CURSOR WITH HOLD FOR HURS01`). Only at transaction end does the DB2 Gateway close the cursor.

Considerations

- A COMMIT request sent to the Gateway or a normal end of TIBCO Object Service Broker transaction results in an SQL COMMIT being sent to DB2.
- A ROLLBACK request sent to the Gateway or a transaction failure results in an SQL ROLLBACK being sent to DB2.
- Intermediate ROLLBACKs are allowed provided cursor positioning is not compromised.
- Processing can continue after a ROLLBACK, however, cursors still open are closed and the application must re-establish the cursor.

The exception COMMITLIMIT does not apply to DB2 tables. Requests to update DB2 data are processed as they are encountered and are not buffered in the intent list.

Updating DB2 Data Only

Transactions that update only DB2 data are recoverable under DB2.

Updating TIBCO Object Service Broker and DB2 Data

TIBCO Object Service Broker provides a method of ensuring data integrity when a TIBCO Object Service Broker transaction updates both DB2 and TIBCO Object Service Broker data in the same transaction. This method is referred to as Fail Safe level-1 processing.

If you did not request Fail Safe processing, transactions that update both DB2 and TIBCO Object Service Broker data can result in discrepancies if the Gateway or the Data Object Broker abnormally terminate during transaction end processing. Refer to Implementing Fail Safe Processing on page 53 for more information.

When a ROLLBACK request is sent to the Gateway, an SQL ROLLBACK is sent to DB2 and the TDS Intent List is discarded.

Static SQL

Using Static SQL for your application enables DB2 accesses to be done more quickly using less resources. When your application is complete, you can collect your TIBCO Object Service Broker DB2 access statements and use them to generate Static SQL. To do this, your system administrator must bring up an Execution Environment with the DB2LOG Execution Environment parameter set to Y. Refer to Chapter 3, Using Static SQL, on page 61 for more information.

See Also *TIBCO Object Service Broker Parameters* for further information on parameters.

Collecting TIBCO Object Service Broker DB2 Access Statements

When collecting TIBCO Object Service Broker DB2 access statements to generate Static SQL, ensure that the following applies:

- If your application includes rules that contain conditions, invoke all statements in your application that access DB2 data. This is done by running the application, perhaps several times, to include all accesses. If an access to a DB2 table is already logged, it is not logged again.
- If a table is accessed by more than one application, you must run all these applications.
- If a DB2 column accepts nulls, and your application is selecting this column for both nulls and values, you must collect access statements for both nulls and values. This is because DB2 requires different SQL statements for nulls than for values.

- If you modify your application, you must rerun the application to collect new TIBCO Object Service Broker DB2 access statements.
- If you modify only your TIBCO Object Service Broker DB2 table definition, you have to regenerate Static SQL only to pick up any changes. You do not have to recollect the TIBCO Object Service Broker DB2 access statements. Refer to Task C: Generate Static SQL Using the @STATICSQL Tool on page 65 for more information.

  Since the Table Editor and the Table Browser are TIBCO Object Service Broker applications, accesses (SELECTs) made through these applications against TIBCO Object Service Broker DB2 tables are also logged.

Error Handling

Gateway Exceptions

The TIBCO Object Service Broker runtime environment signals system exceptions to enable an application to recover from an error. A three-level hierarchy of exceptions exists. Each exception traps the exceptions that appear below it in the hierarchy. All errors encountered when accessing DB2 data through the Gateway are trapped under one of the following exceptions:

ERROR	An error is detected and no lower-level exception exists in the application.
ACCESSFAIL	A table access error is detected.

GETFAIL	No occurrence satisfies the selection criteria. SQL code +100 raises this exception.
DELETEFAIL	The primary key specified for a DELETE statement does not exist. SQL code +100 raises this exception.
INSERTFAIL	The primary key provided for an INSERT statement already exists. SQL error code -803 raises this exception.
REPLACEFAIL	The primary key provided for a REPLACE statement does not exist. SQL code +100 or the SQL error code -803 both raise this exception.

INTEGRITYFAIL	An attempt to violate integrity is detected.

LOCKFAIL	There is a lock on an occurrence or a table is unavailable. SQL error codes -904, -913, and -30040 all raise this exception.
SECURITYFAIL	Permission for the requested action on the TIBCO Object Service Broker object is denied. This also occurs if: The DB2 authorization ID does not have permission to perform the requested action on the specified object. SQL error codes -551, -552, -553, -554, -555, and -556 all raise this exception. The external security interface is implemented and one of the checks performed by the HRNSECD2 macro supplied by TIBCO Object Service Broker failed. Refer to Implementing External Security on page 52 for more information. The external security interface is implemented and the EXTERNALUSERID parameter is set to GROUP and is greater than eight characters.

SERVERBUSY	A new transaction requested an instance of the 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 Gateway made a request to DB2 and DB2 returned 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). Refer to @SERVERERROR Tool on page 124 for more information.
SERVERFAIL	A transaction was in progress when the connection to an instance of the Gateway was broken or the Gateway failed. Control is passed back to the rule for transaction cleanup. SQL error codes -911, -918, -919, -929, -30020, -30030, -30041, -30072, and -30073 also raise this exception.

See Also *TIBCO Object Service Broker Programming in Rules* about exception handling.

@SERVERERROR Tool

Using @SERVERERROR

You must pass @SERVERERROR the contents of RETURN_MESSAGE, which has the following format:

*ppp*AD*nnnx serverid serveruserid source: Message*

The following list describes the variables necessary to pass the RETURN_MESSAGE contents to @SERVERERROR:

ppp	The user-specified 3 character product ID.
nnn	The DB2 external message number.
x	The message severity (E for error, W for warning, and I for Information).
serverid	The server ID of the Gateway.

serveruserid	The Gateway user ID (IDPREFIX + ###) of the Gateway.
source	The code portion of the Gateway that trapped the error and returned the message (for example, CSECT, rule, or function).
Message	The actual error message text.

If a specific message from a specific Gateway has some information that is required to process the error, the table driven approach to the execution of @SERVERERROR causes a rule (specified for that error by the developer using @SERVERERROR) to execute. The error message is interpreted in the @SERVERERROR processing and put into a temporary table until required.

To customize error handling, you must update data in one of the control tables @DB2SQLMSGCNTL or @DB2CAFMSGCNTL. The definition of these tables is owned by TIBCO Object Service Broker and must not be modified. The data you update in them is owned by you.

Table Processing

When the SERVERERROR exception is raised and the @SERVERERROR rule is called by your application, the following steps take place:

1. @SERVERERROR reads the @SERVERMSGCNTL table and looks up the specific message identifier handlers.
2. The appropriate message handler looks up the external error codes in the correct Gateway control tables.
3. If any codes are found, they call the associated user-written handler.
4. The user-written handler can use other functions and data stored in specific tables to handle specific external error/status code.

Considerations

@SERVERERROR can be called at any time, although it is useful only for parsing TIBCO Object Service Broker DB2 messages generated due to external DB2 errors. The original message can always be retrieved using @SE_MSG when @SERVERERROR has been called. The information parsed by @SERVERERROR has transaction scope.

You can add your own instances in the @SERVERMSGCNTL table, provided that the OWNER specified begins with letters A to Z and the key values in their instance are message identifiers in the form D2*nnnx* mentioned in Using @SERVERERROR on page 124.

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

Stored Procedure Processing

In TIBCO Object Service Broker, you CALL a stored procedure table mapping (TIBCO Object Service Broker table of type DB2, subtype P) from a rule. The Gateway sets up a DB2 SQLDA as per the columns in the table definition and issues the following:

```
EXEC SQL CALL :procedure USING DESCRIPTOR :sqlda
```

On successful return (SQLCODE +466), the Gateway builds a row from all IN, INOUT and OUT parameters and returns. The SQLDA also contains the number of result sets that the stored procedure has generated. This information is returned in the #RS# field of the table.

If the stored procedure has generated any result sets, the field @HANDLE@ will be set to a non-zero value denoting the context of this particular call. That context remains available until discarded, either explicitly via a @DB2PROCDISCARD call or implicitly at the transaction end time. The caller is expected to fetch a result set through the FORALL statement against the predefined result-set mapping (TIBCO Object Service Broker table of type DB2, subtype R), which is parameterized by the @HANDLE@ available upon the CALL statement. The cursor-name attribute of this table denotes the particular result set to be fetched.

If the stored procedure reports multiple result sets, you can fetch them in any order. However, only one FORALL statement is possible at one time: A current FORALL against a result set is closed—and the respective result set discarded—by a subsequent FORALL against another result set.

When the Gateway receives a transaction end it closes any result sets and issues a SQL COMMIT.

Appendix A Documenting DB2 Tables

This appendix describes how to document TIBCO Object Service Broker DB2 tables.

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 DB2 table, press PF2 from the Table Definer. The Documentation screen associated with the EMPLOYEE table is shown below.

```
DESCRIPTION OF TABLE            EMPLOYEE                    UNIT: USR40

MODIFIED ON 20 JAN 2000 BY ACC            CREATED ON 02 JAN 2000 BY USR40

KEYWORDS: EMPLOYEE
SUMMARY : DB2 TABLE CONTAINING EMPLOYEE DATA

                               DESCRIPTION
_ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_ This table contains all information on current employees.




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. Complete these fields as follows:

KEYWORDS	Type individual words that briefly describe the table. These words are used by the TIBCO Object Service Broker Keyword Search facility. 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 function keys are supported from the Documentation screen:

PF1	Displays corresponding help for the current field or screen.
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

R

S

T

U

V

W

X