TIBCO® Object Service Broker

Getting Started

Software Release 6.0
July 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

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

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

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

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

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

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

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

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

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

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

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

Contents

Preface

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

This manual provides basic concepts and principles of TIBCO Object Service Broker, an introduction to the terminology associated with the product, an understanding of the fundamental product objectives and architectural concepts. It also includes a description of how to log on, how to use the default developer's workbench, a tutorial and a glossary.

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 TIBCO Object Service Broker Concepts

This chapter provides an overview of TIBCO Object Service Broker.

Topics

What is TIBCO Object Service Broker?

TIBCO Object Service Broker provides a powerful and integrated application development and integration brokering environment. TIBCO Object Service Broker runs natively on z/OS and also Windows and Solaris. It includes a dynamic repository, a high level business process rules engine, and a variety of interfaces to mainframe and open systems application code.

The Windows and Solaris platforms are referred to collectively as Open Systems in the rest of this manual.

Application Development Environment

The primary facilities for application development include the following:

- Database
- Programming language
- Query and reporting facilities
- Developer's workbench
- Administrator's workbench

Integration Broker

TIBCO Object Service Broker acts as an integration broker between legacy and non-legacy databases and applications.

- Its technology gateways and servers provide full access to CICS and IMS TM.
- Its data gateways provide full access to VSAM and sequential files, CA-IDMS, DB2, IMS/DB, Oracle, and data accessible through ODBC. Access to Adabas and CA-Datacom is provided by way of an SDK.
- Its open interfaces include Java Connector Architecture (JCA), JavaBeans, Java Servlet 2.0, COM and .NET, WebSphere MQ, and ODBC, as well as providing XML capabilities.
- It integrates directly with both TIBCO Enterprise Message Service™ (EMS) and TIBCO BusinessWorks™.

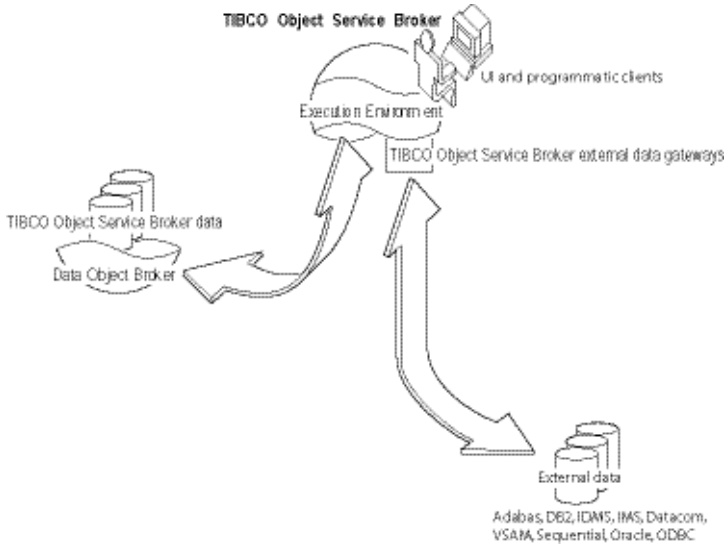
TIBCO Object Service Broker Components

These are the main components of an TIBCO Object Service Broker system:

- Data Object Broker: data repository and management
- Execution Environment: program execution, including access to external data sources
- Client interfaces: text-based and graphical workbenches, and programmatic interfaces

Primary Components Illustrated

The following diagram illustrates the primary components of TIBCO Object Service Broker:



Data Object Broker

The Data Object Broker handles the coordination and management of data. It updates and provides data integrity for the MetaStor and acts as a server to the Execution Environment. It also performs data accesses and updates on behalf of a transaction running in a session.

MetaStor

The MetaStor is an active store for all TIBCO Object Service Broker metadata: definitions, characteristics, access paths, and storage locations of all objects (data and programs) in a node with its own MetaStor. Every Data Object Broker has its own MetaStor. A node is the name associated with a Data Object Broker. In a distributed environment the node name is used to identify a Data Object Broker to another Data Object Broker.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating*, *TIBCO Object Service Broker for Open Systems Installing and Operating*, and *TIBCO Object Service Broker Application Administration* about the Data Object Broker, nodes, and distributed environments.

Execution Environment

The Execution Environment is the component of TIBCO Object Service Broker that interacts with the end user or TIBCO Object Service Broker application. It uses a rules language to manage tasks such as presentation, command interpretation, and logical data control. The Execution Environment runs applications written in rules for the user and communicates with a Data Object Broker on behalf of the user.

Rules Language

The rules language is a very simple programming language used to develop TIBCO Object Service Broker applications. The following lists a few of its characteristics:

- It contains only 17 verbs and 8 data types
- It promotes data-driven processing
- Data access statements are the same regardless of data location
- Data repositories can change without requiring changes to rules code
- Trigger and validation rules can be associated with any table
- System exceptions are supplied and users can define their own exceptions
- The rules code is actually data in a table. It is stored as byte code in an active repository known as the MetaStor

A simple tutorial illustrating how to write applications using the rules language is available in Chapter 3, A Quick TIBCO Object Service Broker Tutorial, on page 37.

See Also *TIBCO Object Service Broker Programming in Rules* for details about the rules language and rules programming.

TIBCO Object Service Broker for z/OS External Environments about the available z/OS programmable interfaces and how to use them.

TIBCO Object Service Broker for Open Systems External Environments about the available programmable interfaces for the Windows and Solaris platforms and how to use them.

Client Interfaces

The client services layer provides the interface between TIBCO Object Service Broker and its host operating environment. Depending on the client interface, users are provided with either an application program interface or a workbench of TIBCO Object Service Broker tools. The following table describes the client interfaces that you can use to access TIBCO Object Service Broker:

Name	Data Object Broker Platform	Function	Default User Interface
SDK (Java)	All	Application Programming Interface (API) that connects TIBCO Object Service Broker and client programs written in Java.	Application program
SDK (C/C++)	All	API that connects TIBCO Object Service Broker and client programs written in third-generation languages, such as C.	Application program
TIBCO Object Service Broker UI	All	Program with a graphical user interface that you use to develop applications.	Workbench
ostty	Open Systems	Program that starts a TIBCO Object Service Broker workbench in a text format.	Workbench
Telnet 3270	Open Systems	Terminal protocol by which TIBCO Object Service Broker can be accessed.	Workbench
osBatch	Open Systems	Program that starts TIBCO Object Service Broker as a batch process.	Application program

Name	Data Object Broker Platform	Function	Default User Interface
S6BTSO	z/OS	Program that starts TIBCO Object Service Broker workbench as a TSO process.	Workbench
S6BCS*xxx*[a]	z/OS	Set of programs that start a TIBCO Object Service Broker workbench as a CICS process.	Workbench
S6BIM*xxx*[b]	z/OS	Set of programs that start a TIBCO Object Service Broker workbench as an IMS TM process.	Workbench
Call Level Interface	z/OS	API that connects TIBCO Object Service Broker and client programs written in third-generation languages, such as COBOL or assembler.	Application program
S6BBATCH	z/OS	Program that starts TIBCO Object Service Broker as a batch process.	Application program
VTAM LU2	z/OS	Connectivity that enables TIBCO Object Service Broker to run as a VTAM application.	Workbench

a. Requires the Service Gateway for CICS, which is a separately purchased component.

b. Requires the Service Gateway for IMS TM, which is a separately purchased component.

See Also *TIBCO Object Service Broker for z/OS External Environments* about the available z/OS programmable interfaces and how to use them.

TIBCO Object Service Broker for Open Systems External Environments about the available programmable interfaces for Open Systems and how to use them.

TIBCO Object Service Broker Architecture

TIBCO Object Service Broker is a collection of components, Data Object Broker, Execution Environment, and client interfaces, based around a dynamic repository known as the MetaStor. It includes interfaces to mainframe and Open Systems application code and transaction environments.

Business transactions can be specified using:

- TIBCO Object Service Broker's high-level rules language
- An external process written in assembler, COBOL, PL1/1, CICS, C, etc.

SDKs are available for use with 3rd generation programming languages including Java, C, and C++.

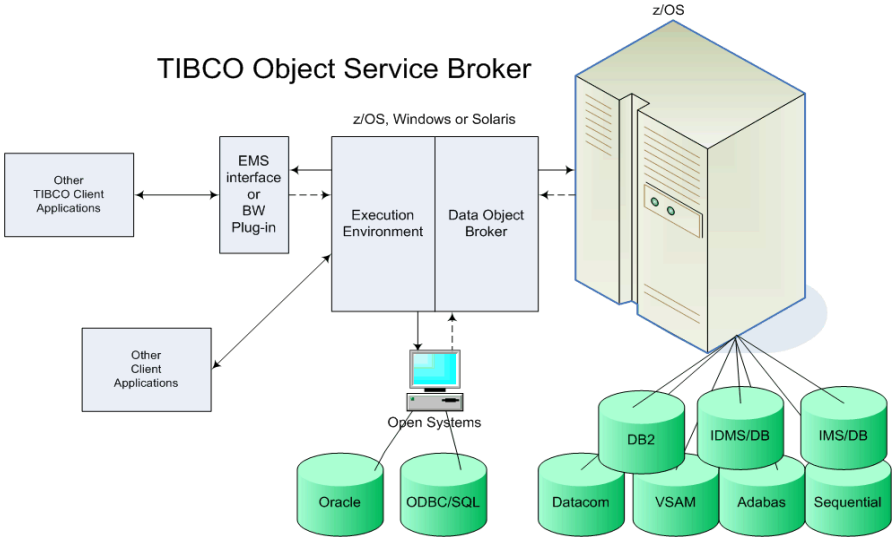
Gateways and Servers

Using supplied gateways and servers, you can bi-directionally access the following data sources: Adabas, CA-IDMS, CA-Datacom, DB2, IMS/DB, ODBC-accessible, VSAM, and sequential data via supplied gateways. The Data Object Broker, gateways, and servers can reside on different domains and operating systems, but the gateway or server must be in the same domain as its data source.

Configuration

TIBCO Object Service Broker's components can be deployed on a single machine or across multiple machines running under z/OS, Windows, and Solaris. A supplied Eclipse-based UI operates on Windows and Solaris and can access a Data Object Broker running on any supported platform.

Architectural Diagram



TIBCO Object Service Broker Integration with other TIBCO Products

TIBCO Object Service Broker integrates directly with both TIBCO Enterprise Messaging Service (EMS) and TIBCO BusinessWorks. The same integration mechanisms are used regardless of whether TIBCO Object Service Broker is running on z/OS, Windows or Solaris.

These interfaces provide access to the native TIBCO Object Service Broker data store and processing environment as well as to all the external data sources available to TIBCO Object Service Broker: DB2, CA-Datacom, CA-IDMS, IMS, Adabas, Oracle, ODBC-compliant data, and sequential and VSAM data.

Integration with TIBCO Enterprise Messaging Service

The interface to EMS provides a set of tools, S6BCALL and S6BFUNCTION, that TIBCO Object Service Broker rules applications running on z/OS and Open Systems can use to produce and consume messages. These messages are transported via TIBCO EMS servers which run on many platforms.

The first call to EMS by a rule initializes the environment to run EMS and loads code related to invoking EMS. The types of arguments and the return value are determined by the EMS C routine being invoked.

See Also *TIBCO Object Service Broker for z/OS External Environments* and *TIBCO Object Service Broker for Open Systems External Environments* about using the EMS interface.

Integration with TIBCO BusinessWorks

The TIBCO BusinessWorks OSB Plug-in is supplied with TIBCO Object Service Broker. The plug-in allows you to retrieve and send TIBCO Object Service Broker data in a predefined manner so that it can take part in BusinessWorks processes.

An OSB palette is available from within TIBCO Designer and you can use the following resources from within the palette:

- **OSB Connection resource** This shared configuration resource specifies the connection details to access the TIBCO Object Service Broker transactions: host, port, user ID, password, and so on.
- **OSB Transaction resource** This shared resource represents an available TIBCO Object Service Broker transaction and contains its XML schema. You can use it across multiple BusinessWorks activities to feed data to the transaction and access the transaction result data.

- **Invoke an OSB Transaction activity** This activity sends a request to invoke a specified transaction by passing the inputs, getting the response, and parsing the results as XML to make it available as output to other BusinessWorks processes

See Also *TIBCO BusinessWorks OSB Plug-in User's Guide* about using the plug-in.

Chapter 2 Using the TIBCO Object Service Broker Development Environment

This chapter describes how to log on to TIBCO Object Service Broker and get started with the development environment.

Topics

Overview

Installed Components

During the installation of TIBCO Object Service Broker, you, or your system administrator, installed on your computer either of the following:

- Only the clients of TIBCO Object Service Broker, with the server components residing on a machine other than your own
- Both the server and the client components of TIBCO Object Service Broker

If the server components reside on another machine, you must ensure that TIBCO Object Service Broker is installed and started before you can start to use the development interfaces.

Development Interfaces

The TIBCO Object Service Broker UI and the text-based workbench provide you with the tools needed for prototyping, testing, and completing the development and implementation of an application.

You can use either the TIBCO Object Service Broker UI or a text-based workbench to define TIBCO Object Service Broker objects such as tables, rules, screens, and reports. The TIBCO Object Service Broker UI is an Eclipse-based graphical user interface that you can use for application development. The text-based workbench is a standard menu that includes definers, editors, a number of shareable tools, and some administrative interfaces.

Basic Tutorial Available

A basic tutorial is available in this manual to take you through the steps and concepts required to develop a TIBCO Object Service Broker application. Refer to Chapter 3, A Quick TIBCO Object Service Broker Tutorial, on page 37.

Starting the Development Interfaces

Tasks to Getting Started

You must complete the following tasks to start to use the development interfaces:

1. Ensure the server components are started if you are accessing TIBCO Object Service Broker on another machine. You may need to check with your System Administrator.

   or

   Start the server components if they reside on your machine. This is described below.
2. Choose your user interface
3. Specify connection properties

Starting the Server Components

On Windows

If the server components reside on your Windows computer, before starting the workbench, you must complete the following procedure.

From the TIBCO Object Service Broker program menu, available under the TIBCO program group:

4. Choose DOB Start.

   This starts a background process for your database.
5. Choose Start osMon.

   This starts a background process for your application development.

On Solaris

If the server components reside on your Solaris computer, before starting the workbench, you must complete the following procedure.

From a command prompt, issue the following commands:

1. hrncr

   This starts background processes for your database.
2. osMon

This starts a process for your application development.

On z/OS

If the server components reside on z/OS, before starting the workbench ensure that the TIBCO Object Service Broker system is running. For details on how to start a system on z/OS refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Choose Your User Interface

You can access TIBCO Object Service Broker using one of the following:

- The TIBCO Object Service Broker UI, which is an Eclipse-based UI.
- A supplied text workbench, ostty, which runs on Open Systems
- A supplied text workbench which runs under TSO on z/OS
- A customer-supplied Telnet 3270 emulation program, which is used to display a text workbench on Open Systems

The following sections describe how to get started with ostty, a TSO managed text workbench, and a Telnet 3270 emulation program. For details about getting started with the TIBCO Object Service Broker UI, refer to Getting Started with the TIBCO Object Service Broker UI on page 18.

Text workbenches are also available under CICS and IMS TM. These require the appropriate Service Gateways to be installed which are separately purchased components of TIBCO Object Service Broker.

Predefined User IDs and Passwords Supplied

TIBCO Object Service Broker comes with three predefined user IDs and passwords to get you started. The default user ID at installation time is SYSADMIN. You use the Security interface to define other user IDs and passwords.

User ID	Password
SYSADMIN	SYSADMIN
HURON	HURON
HURON1	HURON1

See Also | *TIBCO Object Service Broker Managing Security*

Starting an ostty Session

To start an ostty session, complete the following:

1. On the Start menu, go to the TIBCO Object Service Broker program menu and select ostty.
2. At the displayed prompt, enter the password for the specified User ID. The password and user ID are case sensitive.
3. Press Enter.

   This displays a text-based workbench. For details on how to use this workbench, refer to Understanding the Default Developer's Workbench on page 22.

Starting a TSO Session

The USER CLIST, distributed with TIBCO Object Service Broker in the CLIST data set, runs a TIBCO Object Service Broker TSO client. The list of parameters that you can specify when you start your session—and the default values for some of the parameters—should be customized for your installation usage.

The following shows how to execute the USER clist under ISPF 6:

```
exec '$HLQNONV$.$INSTVER$.clist(user)' 'u(huron1) p(huron1)'
```

See Also For more details about starting a TSO session, refer to *TIBCO Object Service Broker for z/OS External Environments*.

Starting a VTAM LU2 Session within a Native Execution Environment

VTAM LU2 sessions are established by logging in through a VTAM LU2 (3270) terminal. To start a VTAM LU2 session, use the following command:

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

where:

vtamapplid	The EENAME parameter value specified when the Native Execution Environment was initialized. Ask your system administrator to provide you with this value.
userid	A valid TIBCO Object Service Broker user ID.
password	[optional requirement] The password of the user ID set by the U parameter.

See Also For more details about starting a VTAM LU2 session within a Native Execution Environment, refer to *TIBCO Object Service Broker for z/OS External Environments*.

Starting a Telnet 3270 Session

To start a session using a customer-supplied 3270 terminal emulator, complete the following:

1. On the Start menu, start your Telnet 3270 application.

   Enter the name of the host where you want to connect to.
2. This is the machine name or the IP address where osMon is running.
3. Enter the number of the port where osMon is listening.

   Default value is 9099.
4. Select the 3270 type and the 3270 model you want to emulate.

   3278 and 3279 types of terminals, using models 2, 3, 4 and 5 are supported.
5. Click the appropriate button to connect to the Execution Environment.

   The TIBCO Object Service Broker login screen appears.
6. Enter your TIBCO Object Service Broker user ID and password (refer to the note after step 8). The password and user ID are case sensitive.

   These are limited to a maximum of eight characters each.
7. [Optional] Enter any session parameters you want to override.

   You can leave this field blank. You enter information in this field only if you want to override the values for the session parameters. Any information you enter is limited to 128 characters over 2 lines.
8. Press Enter.

   Press Enter to display a text-based workbench.

- When logging in to TIBCO Object Service Broker using Telnet 3270, your password is sent over the network as clear text.

For details on how to use the text-based workbench, refer to Understanding the Default Developer's Workbench on page 22.

Customizing Your Session Options

Default values are supplied for client sessions. You can customize these values via session parameters and with the $SETOPT shareable tool. Refer to *TIBCO Object Service Broker Parameters* for details about the parameters available to customize your session and how to change them. Refer *TIBCO Object Service Broker Shareable Tools* for details about using $SETOPT.

Getting Started with the TIBCO Object Service Broker UI

Tasks Required to Get Started

In order to get started you need to:

1. Start Eclipse
2. Select a workspace folder
3. Open an OSB Perspective
4. Create an OSB Project
5. Specify connection settings

Task 1: Start Eclipse

On Windows, from your Start menu, select Eclipse SDK.

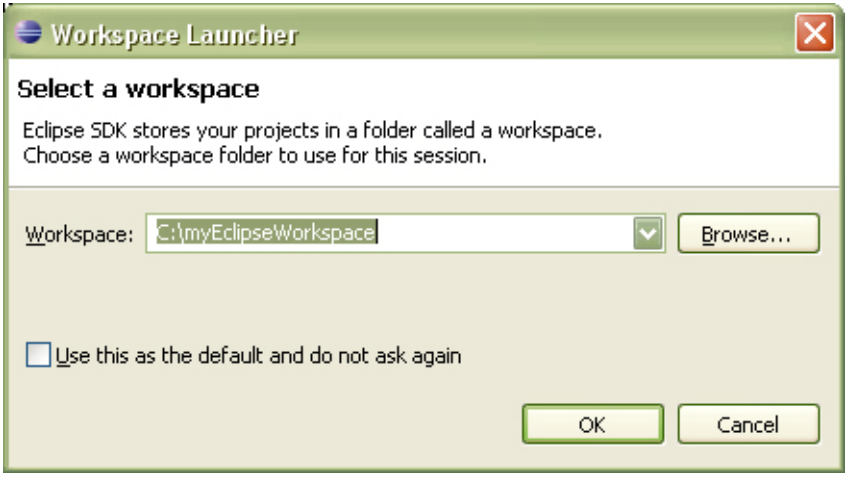
On Solaris, navigate to TIBCO_HOME/eclipse and type eclipse. Press Enter.

In both cases, this displays a Workspace Launcher.

Task 2: Select a Workspace Folder

Using the displayed Workspace Launcher, select a workspace folder. You can create a new folder or use an existing workspace.

The following shows the Workspace Launcher:

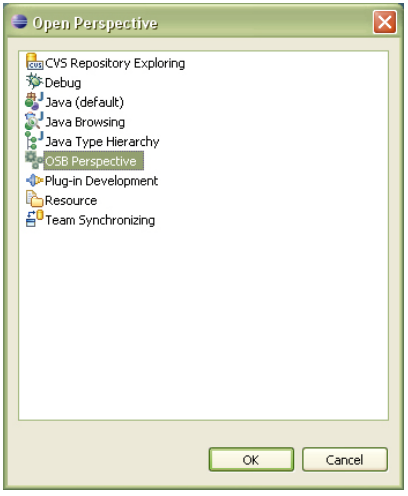


Task 3: Open an OSB Perspective

From the Eclipse window, as shown in the following illustration:

1. Select the Window menu item.
2. Select Open Perspective, from the drop-down menu.
3. Select Other..., from the drop-down menu.

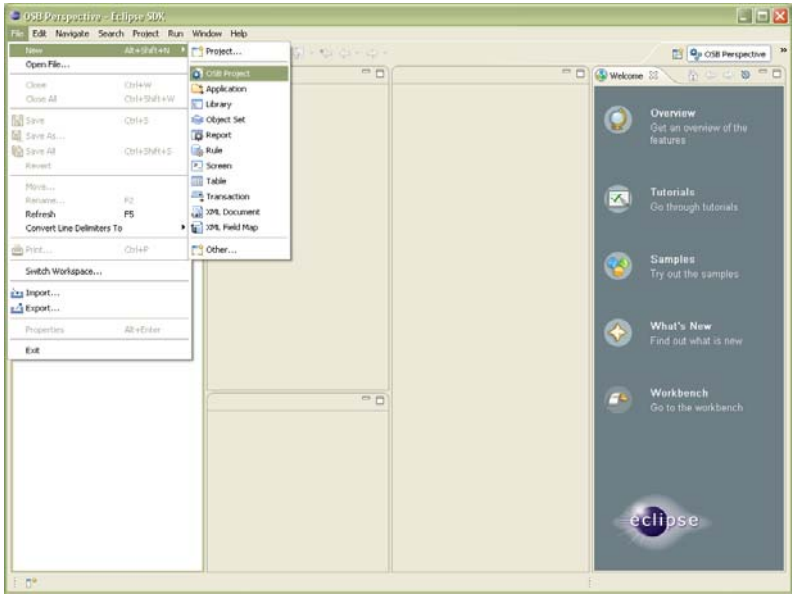
4. From the Open Perspective window, select OSB Perspective as shown in the following illustration:



Task 4: Create an OSB Project

From the OSB Perspective window, as shown in the following illustration:

1. Select the File menu item.
2. Select New, from the drop-down menu.
3. Select OSB Project, from the drop-down menu.



Task 5: Specify Connection Settings

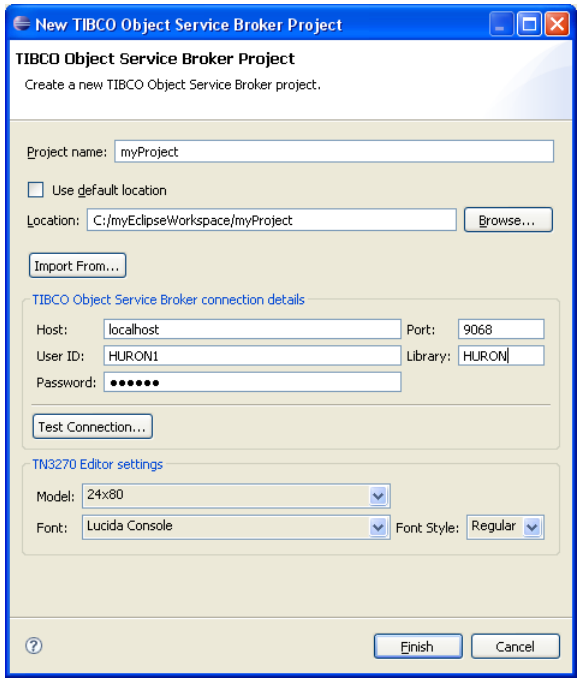
If the server is on your computer, use the following values in the window which is shown in the following illustration:

- User ID: HURON1
- Password: HURON1 (The password field is case-sensitive.)
- Port: 9068
- Host: localhost

If you prefer, you can define and then use your own user ID, password, and library.

Once completed, click test Connection… to verify the connection.

If you are connecting to a server on another computer, you need to find out from your administrator the values to use for host, port, user ID, and password.



You can use the **Import From...** button if you already have a TIBCO Object Service Broker project and would like to reuse connection information and other properties that are applicable to this version of the UI project.

If you encounter errors when using the Eclipse UI, you can refer to the Eclipse error log for more information. To access the error log:

1. Open the Windows tab.
2. Select Show View.
3. Select Error Log.
4. Select the message you wish to view.

You may need to scroll right to view a complete message.

Using the TIBCO Object Service Broker UI

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

Understanding the Default Developer's Workbench

TIBCO Object Service Broker UI

This section describes how to perform various tasks in TIBCO Object Service Broker using the text-based workbench. You can also perform development tasks using the TIBCO Object Service Broker UI, which provides a graphical and text-based environment for TIBCO Object Service Broker development. For information about using the TIBCO Object Service Broker UI, refer to the TIBCO Object Service Broker UI online help.

Default Developer's Workbench

When you first log in to TIBCO Object Service Broker using a text-based client the default developer's workbench appears. The developer's workbench is also used from within the TIBCO Object Service Broker UI.

Refer to Appendix A, Common TIBCO Object Service Broker Text-Based Workbench Functionality, on page 75 for additional information about using this interface.

Customized Workbench

Your system administrator can also customize a workbench for your environment. For details on customizing a workbench, refer to *TIBCO Object Service Broker Defining Screens and Menus*.

Default Developer's Workbench Illustrated

```
                  USR40    TEST: N BROWSE: N  2:54 PM   THURSDAY FEB 22 2007

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

   COMMAND ==>  __


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

Using the Workbench

This section describes how to use the workbench. The following topics are discussed:

- Moving around the workbench—refer to Moving Around the Workbench below for more information
- Using PF keys—refer to Using PF Keys for more information
- Using tools menu commands—refer to Using Tools Menu Commands on page 31 for more information
- Available functionality—refer to What Functionality is Available from the Workbench? on page 26 for more information

If you change the font for your Telnet 3270 program, ensure that you use a fixed pitch (monospace) font, such as Courier New, Lucida Console, Courier, Fixedsys, or Modern, to get a legible representation of the applications.

Moving Around the Workbench

To move around the workbench, use the following control keys:

Key	3270 and 3270 Emulators
Tab	Moves the cursor to the beginning of the next row.
Up Arrow	Moves the cursor up one row.
Down Arrow	Moves the cursor down one row.
Left Arrow	Moves the cursor to the left one character or field.
Right Arrow	Moves the cursor to the right one character or field.
Home	Moves the cursor to the **Library** field (the first unprotected field).
End	Moves the cursor to the end of the current row.
Ctrl+Home>	Moves the cursor to the **Library** field (the first unprotected field) and clears all unprotected fields on the screen.
Ctrl+End	Clears the content of the field where the cursor is sitting from the cursor to the end of the field inclusively.

In a Windows environment, depending on your 3270 emulator, the keys PF1, PF24, and Enter does or does not reset the insert keyboard mode.

Using PF Keys

The following PF keys are available from the z/OS, Open Systems workbenches. For more information on the available PF keys, place the cursor on the screen outside of a field and press PF1. For more information on standard PF keys, refer to Appendix A, Common TIBCO Object Service Broker Text-Based Workbench Functionality, on page 75.

PF Key	Function
PF1	Help
PF2	Message log
PF3	Exit
PF7	Scroll up
PF8	Scroll down
PF12	Exit
PF24	Refresh

What Functionality is Available from the Workbench?

The default workbench is made up of the following components:

- Common session attributes
- The supplied tools
- Messaging mechanisms

These components are described in the following sections.

Displayed Session Attributes

The following session attributes appear at the top of the default workbench:

- **Library** field
- **Test** field
- **Browse** field

Library Field

When you first enter TIBCO Object Service Broker, the **Library** field displays the name of your default library, which is specified in your user profile and which contains the rules that you created and rules from other libraries that you edited and saved, or copied into your library. You can have rights to other libraries and subsequently other rules. You can change to another library to which you have access, by typing over the name of the library. For more information on library management, refer to the *TIBCO Object Service Broker Programming in Rules* manual and the DEFINE_LIBRARY tool in the *TIBCO Object Service Broker Shareable Tools* manual.

Test Field

The **Test** field applies only to the execute rule workbench tool. No other functionality of the workbench is affected. It determines whether a rule, when executed, runs in test mode.

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

Browse Field

The **Browse** field applies only to the execute rule workbench tool. No other functionality of the workbench is affected. This field overrides the workbench specification and determines if a rule, when executed, runs in browse or update mode. Other tools, like the Menu Definer, give you the option of specifying the mode in which your rule executes.

When the **Browse** field is set to Y, you cannot make updates to most types of tables or to external databases, through the execution of a rule. If you attempt to make an update, the rule fails. Export (EXP) tables and memory-resident tables are updated.

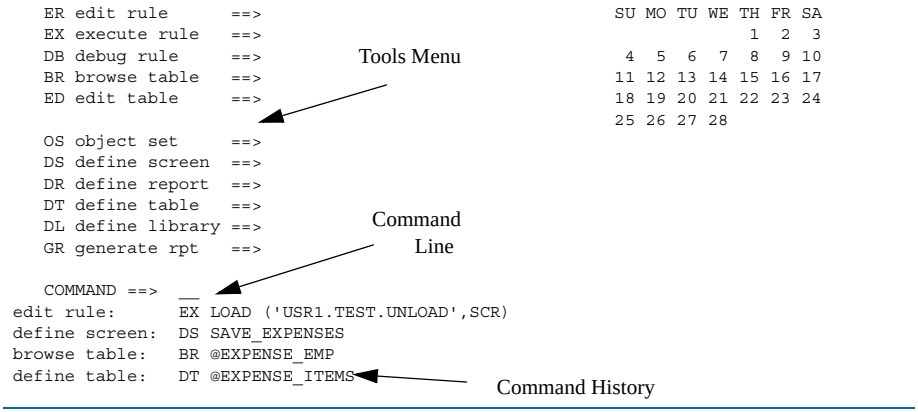
Accessing the Supplied Tools

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

- Tools menu
- Command line
- Command history area

Example

The following figure shows the various entry points into the supplied TIBCO Object Service Broker tools:



Accessing Tools from the Tools Menu

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

1. Tab to place the cursor in the input field on the corresponding line for the tool.
2. Type in an object name, if required.

   For example: `BR browse table ==> EMPLOYEES`

   If the name of an object (rule, table, screen, and so on) is not typed in, the Object Manager screen is invoked when you press Enter. Refer to Managing MetaStor Objects on page 89 for more information on the Object Manager.
3. Type in any required arguments or parameters.

   For example: `BR browse table ==> EMPLOYEES (MIDWEST)`

Rules arguments or table parameters can be supplied with table or rules names. If the necessary arguments or parameters are not supplied, a prompt screen appears when you press Enter.

```
BROWSING TABLE   :  EMPLOYEES


ENTER PARM VALUE REGION          :









PFKEYS: 1=HELP 3=EXIT 12=EXIT
```

4. Press Enter.

Accessing Tools from the Command Line

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

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

1. Place the cursor on the command line.
2. Type in the appropriate command abbreviation.
3. Type in an object name, if required.

   If the name of an object (rule, table, screen, report, and so on) is not provided, the Object Manager screen is invoked when you press Enter. Refer to Managing MetaStor Objects on page 89 for more information on the Object Manager.

4. Type in any required arguments or parameters.

   Rules arguments or table parameters can be supplied with table or rules names. If these are not supplied, a prompt screen appears when you press Enter.

5. Press Enter.

Accessing Tools from the Command History Area

The command history area displays previously executed commands. This scrollable list can display the last 50 commands issued. To scroll the information vertically, move the cursor into the area and press PF7 and PF8.

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

1. Place the cursor on the appropriate command.
2. Make any modifications to the command by typing over existing text.

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

3. Press Enter.

Using Tools Menu Commands

The following table lists the tools that appear on the default workbench and provides a reference to where you can find more information on each tool.

Workbench Tool	Reference
ER edit rule	*TIBCO Object Service Broker Programming in Rules*
EX execute rule	*TIBCO Object Service Broker Programming in Rules*
DB debug rule	*TIBCO Object Service Broker Programming in Rules*
BR browse table	*TIBCO Object Service Broker Managing Data*
ED edit table	*TIBCO Object Service Broker Managing Data*
OS object set	*TIBCO Object Service Broker Application Administration*

Workbench Tool	Reference
DS define screen	*TIBCO Object Service Broker Defining Screens and Menus* *TIBCO Object Service Broker Application Administration*
DR define report	*TIBCO Object Service Broker Defining Reports*
DT define table	*TIBCO Object Service Broker Managing Data* *TIBCO Object Service Broker Managing External Data* *TIBCO Object Service Broker Application Administration*
DL define library	*TIBCO Object Service Broker Programming in Rules*
GR generate report	*TIBCO Object Service Broker Defining Reports*
CD copy definition	*TIBCO Object Service Broker Shareable Tools*
CT copy table	*TIBCO Object Service Broker Shareable Tools*
CL clear table	*TIBCO Object Service Broker Shareable Tools*
DD diff definition	*TIBCO Object Service Broker Shareable Tools*
PR print rules	*TIBCO Object Service Broker Shareable Tools*
PT print table	*TIBCO Object Service Broker Shareable Tools*
SR search utility	*TIBCO Object Service Broker Shareable Tools*
SE security manager	*TIBCO Object Service Broker Managing Security*
PM promotion	*TIBCO Object Service Broker Parameters*
MR manage rights	*TIBCO Object Service Broker Managing Deployment*
UP user profile	*TIBCO Object Service Broker Managing Security*

Messaging Mechanisms

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

- Broadcast message
- Message log

Broadcast Message

An administrator would use the broadcast message table to convey a message to all users when they first log in to TIBCO Object Service Broker. It appears at the bottom of the workbench next to the time, when you first log in to the system.

The message text for the broadcast message is obtained from the BROADCAST table, which is a TIBCO Object Service Broker table. The occurrence with the highest primary key (the NUMBER field) appears on the workbench. You can view the complete table by pressing PF2 before you start another transactions in your current TIBCO Object Service Broker session. If the table is empty, the message “no broadcast available” appears.

BROADCAST Table Illustrated

The following screen shows an occurrence from the BROADCAST table:

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

NUMBER           : 16841
TEXT             : Promotion scheduled for 2007-02-22 at 19:50:15
                 :









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

Message Logs

TIBCO Object Service Broker maintains two message logs that you can view when you press PF2:

- System log
- User log

See Also *TIBCO Object Service Broker Programming in Rules* for information about the message logs.

System Log

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

User Log

The user log contains the messages generated by a rule and the output generated by a transaction (parent) and its descendant (child) transactions. Only output from the most recent transaction plus that from the next higher level are available, because output from the previous lower-level transaction is cleared when a new one begins.

Messages from the system log appear first; pressing PF2 again displays the user log. If the system log is empty, the user log appears after the first time you press PF2.

Chapter 3

A Quick TIBCO Object Service Broker Tutorial

This chapter provides you with a short tutorial to help you get started with developing applications with TIBCO Object Service Broker.

Topics

Tutorial Overview

Purpose of the Chapter

This chapter provides a basic introduction to TIBCO Object Service Broker by demonstrating the steps necessary to build a simple application. This application:

1 Accesses two external data sources to extract information

2. Applies transformations on the data

3. Produces the output according to the specified requirements

The chapter provides enough information to build the application and then refers you to the appropriate manuals for complete information about the tools used.

If you want to actually build the application as you read this chapter, first ensure that you can log in to TIBCO Object Service Broker and that your security permissions allow you to define tables, screens, and rules.

Purpose of the Application

This application will allow the user to access information about games scheduled for the National Hockey League (NHL) season, the scores for games played, and the team standings. The input is provided by two text files:

- One contains the game schedule, stipulated at the beginning of the season
- The other that contains game results and is updated periodically as the season progresses

The application builds the native TIBCO Object Service Broker table used to store the game schedule, processes the results using information directly from the external file, and builds team statistics based on these results. The application also implements business logic to fix, if possible, any record with missing or inconsistent data.

What You will Learn

This simple scenario will illustrate in principle how to use TIBCO Object Service Broker to:

- Access an external data source
- Store data in TIBCO Object Service Broker native data format
- Implement business logic using TIBCO Object Service Broker rules language

- Create output and export it to the text file
- Create a TIBCO Object Service Broker transaction object to invoke processing
- Make processing results available to a TIBCO BusinessWorks process

The application built here does not include the implementation of a User Interface. TIBCO Object Integration Gateway can be used to add the graphical user interface using the environment of your choice (i.e., Java, .NET, VisualBasic) or to invoke TIBCO Object Service Broker processing from the TIBCO BusinessWorks engine.

Task A: Create a Table Definition to Access an External Data Source

TIBCO Object Service Broker stores data in tables. There are a number of different types of tables, with the table type reflecting the data source and the intended use for the table in an application. You define tables using the Table Definer tool.

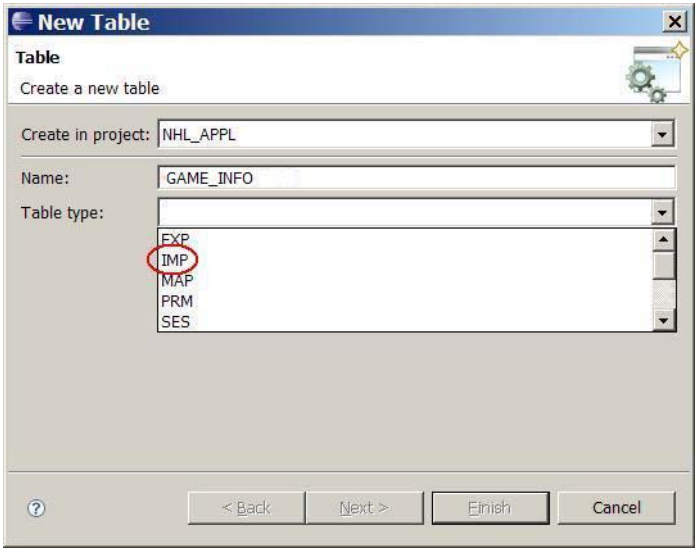
When using an external data source, the basic principle is to make external data look just like TIBCO Object Service Broker data. The programmer writing the application does not need to be aware of the structure of the external database. However, the external data structure must be mapped to a TIBCO Object Service Broker table structure through the Table Definer.

This task has you creating the import (IMP) table that you will need to access the external data source.

Defining an Import (IMP) Table

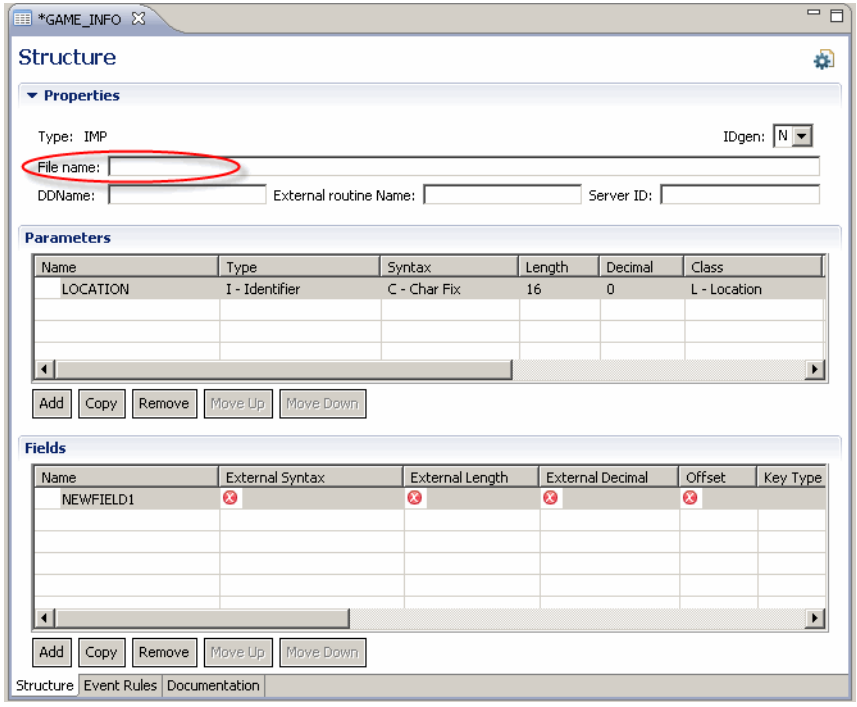
To define a table, complete the following process:

1. Start the New Table Wizard using the File menu and enter the name GAME_INFO for the new table.



Because you are creating a table definition that will import the data from a text file, select IMP, which stands for import, as the table type from the drop-down list.

2. Use the Table Definer panel, which is displayed after pressing Finish, to specify the location and name of the text file, as well as the name and description for each of its fields.



Source for the Games

The information about all NHL games for the 2006-2007 season is available from the text file shown here:

About the Data

Each field is described by its properties in the external data source (i.e., external syntax, external length, offset…) as well as by its TIBCO Object Server Broker properties (type, syntax, length…)

As shown in the illustration above:

- The external length field in the table definition reflects the actual string length as it is used in the text file
- The offset field in the table definition reflects the actual field offset from the beginning of the row

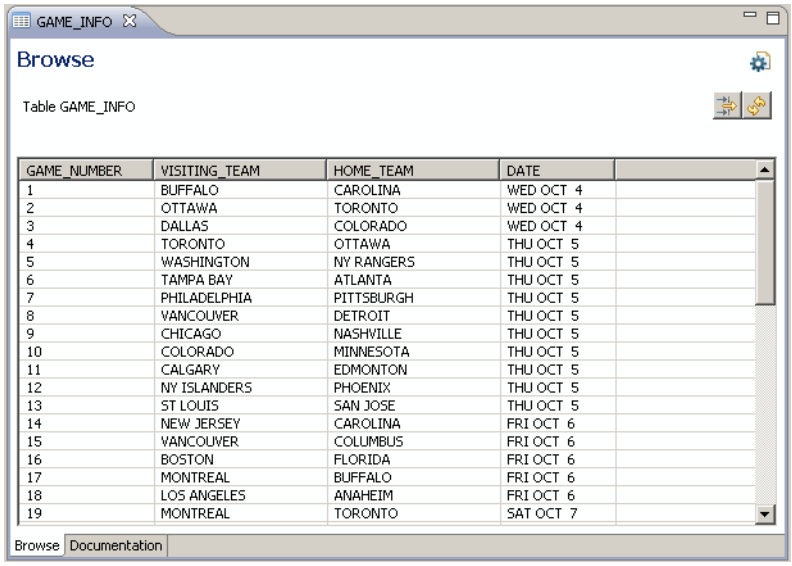
When you input the values for external syntax and length, the Table Definer offers the corresponding values for TIBCO Object Service Broker syntax and length of the field; you can accept the suggested values or change them.

This is the definition of the fields in the Table Definer:

Fields

Name	External Syntax	External Length	External Decimal	Offset	Key Type	Type	Syntax	Length	Decimal
GAME_NUMBER	C - Fixed length character string	4	0	0	P - Primary	C - Count	P - Packed	5	0
VISITING_TEAM	C - Fixed length character string	15	0	17		S - String	C - Char Fix	15	0
HOME_TEAM	C - Fixed length character string	15	0	38		S - String	C - Char Fix	15	0
DATE	C - Fixed length character string	10	0	5		S - String	C - Char Fix	10	0

After creating and saving the definition, you can use the Table Browser to see the contents of the file from within the TIBCO Object Service Broker UI:



GAME_NUMBER	VISITING_TEAM	HOME_TEAM	DATE
1	BUFFALO	CAROLINA	WED OCT 4
2	OTTAWA	TORONTO	WED OCT 4
3	DALLAS	COLORADO	WED OCT 4
4	TORONTO	OTTAWA	THU OCT 5
5	WASHINGTON	NY RANGERS	THU OCT 5
6	TAMPA BAY	ATLANTA	THU OCT 5
7	PHILADELPHIA	PITTSBURGH	THU OCT 5
8	VANCOUVER	DETROIT	THU OCT 5
9	CHICAGO	NASHVILLE	THU OCT 5
10	COLORADO	MINNESOTA	THU OCT 5
11	CALGARY	EDMONTON	THU OCT 5
12	NY ISLANDERS	PHOENIX	THU OCT 5
13	ST LOUIS	SAN JOSE	THU OCT 5
14	NEW JERSEY	CAROLINA	FRI OCT 6
15	VANCOUVER	COLUMBUS	FRI OCT 6
16	BOSTON	FLORIDA	FRI OCT 6
17	MONTREAL	BUFFALO	FRI OCT 6
18	LOS ANGELES	ANAHEIM	FRI OCT 6
19	MONTREAL	TORONTO	SAT OCT 7

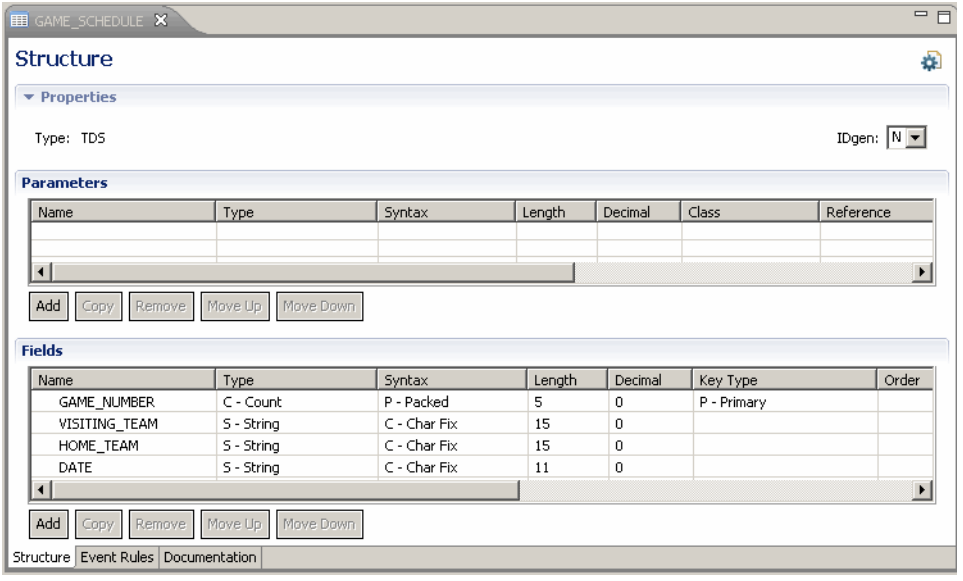
By creating the table definition, the contents of the external data source are made available to TIBCO Object Service Broker. You can now use the rules language to store it internally, analyze it, transform it, etc.

Task B: Create a Table Definition to Store Data in Native TDS Table Format

In this task you will create a table definition so that the games schedule can be stored as a TDS table, which is the native table type of TIBCO Object Service Broker. You can then store the data, and if required transform it using the rules language.

Definition of the TDS Table

Using the Table Definer, create a TDS table called GAME_SCHEDULE with a definition as shown in the following illustration:



Introduction to Rules

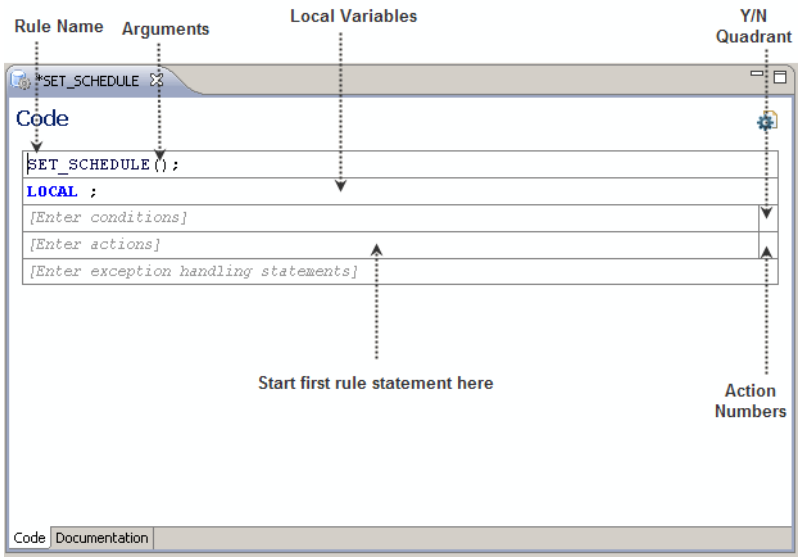
TIBCO Object Service Broker includes a high-level programming language called the rules language that you use to create and modify applications. You use the Rule Editor, a tool available from the workbench, to write the rules that drive your application. TIBCO Object Service Broker stores rules in a rules library; for this application, you can use your default library (displayed at the top of the workbench) to store your rules.

Supplied Programs: Shareable Tools

TIBCO Object Service Broker also includes a large number of supplied programs, called shareable tools, which expedite programming and application development. Shareable tools perform common TIBCO Object Service Broker functions such as string manipulation, mathematical calculation, and object handling.

Format of a Rule

The Rule Editor appears as follows, with the sections as shown:

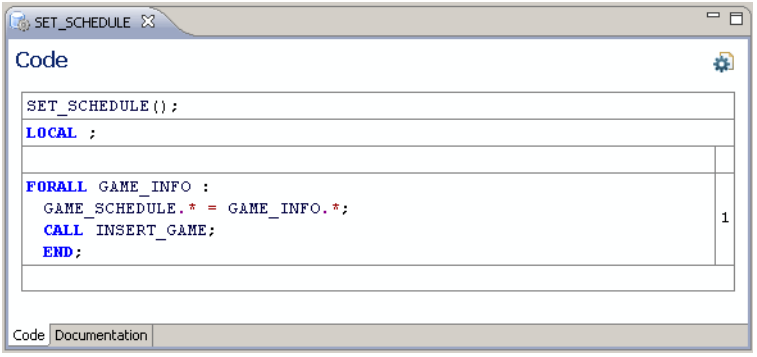


Task C: Create Rules to Populate the TDS Table using Data from the External Data Source

In this task you will create a rule called SET_SCHEDULE that populates the GAME_SCHEDULE table, using information imported from the GAME_INFO table. You will also create a rule called INSERT_GAME that inserts the data into the database.

Populating a Table

The SET_SCHEDULE rule used to populate the table is as follows:



About this Rule

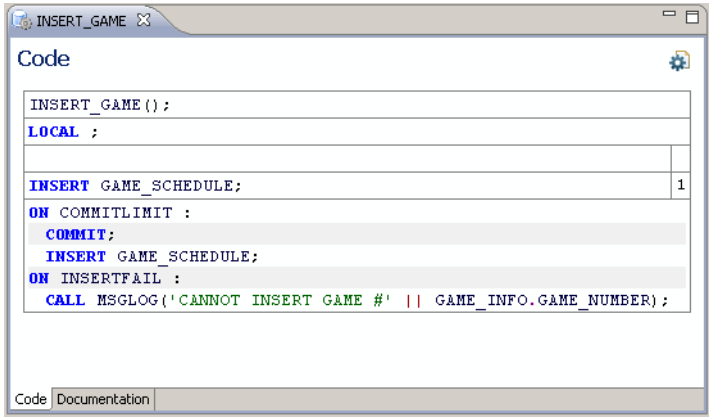
A FORALL statement, which is used here, is a looping construct in the rules language. The entire FORALL loop only has one action, as shown by the Action Number "1", associated with it. Consequently, all the statements between FORALL and END statements are repeated for each occurrence (row) in the table GAME_INFO.

Inside the FORALL loop, the assignment statement (tablename.*=tablename.*) assigns the value of each field from the import table GAME_INFO to the field with the same name in the table GAME_SCHEDULE. The fields that do not have the same names will be ignored.

Inserting the Data into the Database

Once the fields are effectively copied from one row in GAME_INFO to the row in GAME_SCHEDULE, another rule, INSERT_GAME, is called to perform an insert into the database.

The INSERT_GAME rule used to insert the data is as follows:



About this Rule

- The only action statement inserts the content of the row as set in the calling rule SET_SCHEDULE into the table GAME_SCHEDULE.
- The INSERT_GAME rule contains two exception handlers:
  - If the insert operation causes the number of outstanding updates on behalf of the transaction to become too big, the updates will be propagated to the database and changes made permanent using the COMMIT statement. For more information about transaction processing and the COMMITLIMIT exception see the *TIBCO Object Service Broker Programming in Rules* manual.
  - If the insert fails, and the INSERTFAIL system exception is raised (such as the row inserted has a duplicate key), the exception is caught inside the inner rule, a message is printed to the TIBCO Object Service Broker message log, and then control is returned to the outer calling rule and processing of the loop continues.

  This illustrates how by the positioning of the exception handler, the programmer directs the processing flow in case of an error: If we wish to stop further inserts when any of the insert operation fails, the exception handler will be placed in the outer SET_SCHEDULE rule.

Task D: Adding Business and Integration Logic: Building Team Statistics

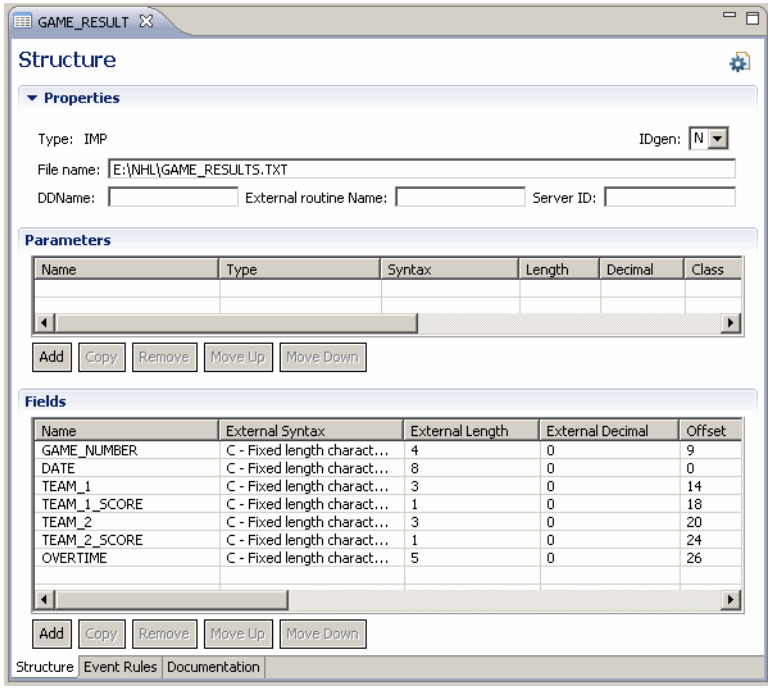
In this task you will access an external file to obtain game results, and then build a list of all the teams showing each team's wins, losses, and goals for and against.

Import File

As the season progresses, the game results are made available via a text file as shown below:

```
200061004 0002 ott 4 tor 1 final
200061004 0001 buf 3 car 2 f/ot
200061004 0003 dal 3 col 2 f/ot
200061005 0009 chi 8 nas 6 final
200061005 0010 col 2 min 3 f/ot
200061005 0012 nyi 3 pho 6 final
200061005 0007 phi 0 pit 4 final
200061005 0013 stl 4 san 5 f/ot
200061005 0005 was 2 nyr 5 final
200061005 0004 tor 6 ott o final
200061005 0006 tam 3 atl 2 f/ot
200061005 0008 van 3 det 1 final
```

The following table definition is used to access this file:



About this Definition

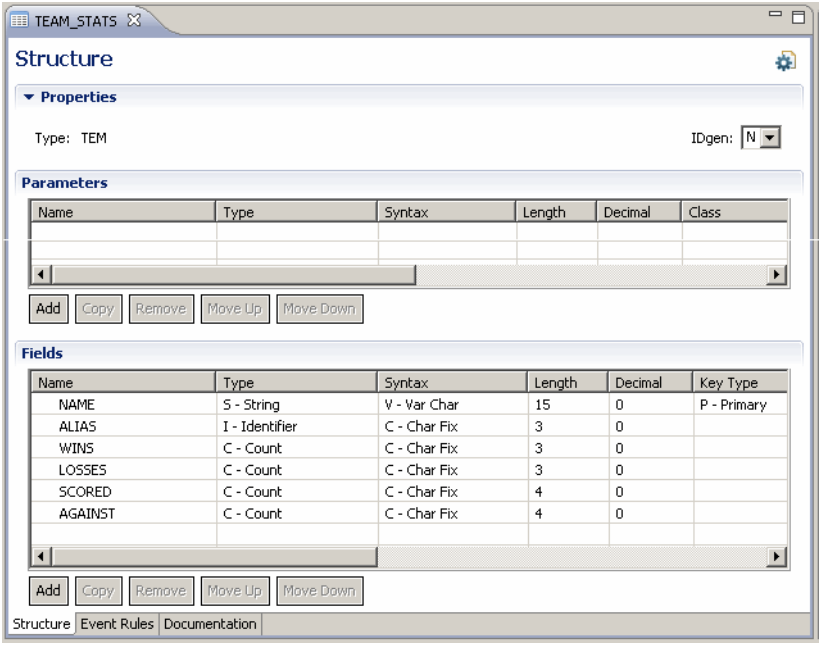
This definition uses the TIBCO Object Service Broker field type C – Count for the fields that will be holding the score (HOME_SCORE, VISITING_SCORE). This is to ensure that arithmetic operations will be permitted with these fields, as we are likely to use those values in addition when calculating the overall number of goals that the team scored.

Reason for Keeping the Source Data External

The number of records coming into the system for the application as described clearly is not overwhelming; it would not cause any problems to store all the results internally and generate statistics using internal information. In the real world, however, we may deal with an external data source that contains millions of records, and importing all of that content to be able to process it would be highly impractical. To illustrate how TIBCO Object Service Broker deals with that scenario, we do not store the results information internally. Instead the application uses the information directly from the external source.

Output Table

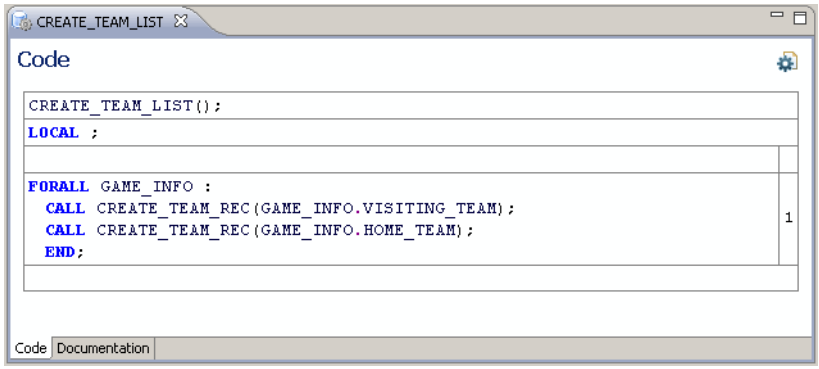
The application output is a table, TEAM_STATS, containing one row per team playing in the league, with information about the overall number of wins, losses, overall number of goals the team has scored, and the overall number of goals against. The output table is defined as a TEM type table, which means that is never made persistent in the database but used to hold temporary data. The content of the TEM table is available within the life span of the database transaction and is discarded when the transaction is done. The table definition is shown below:



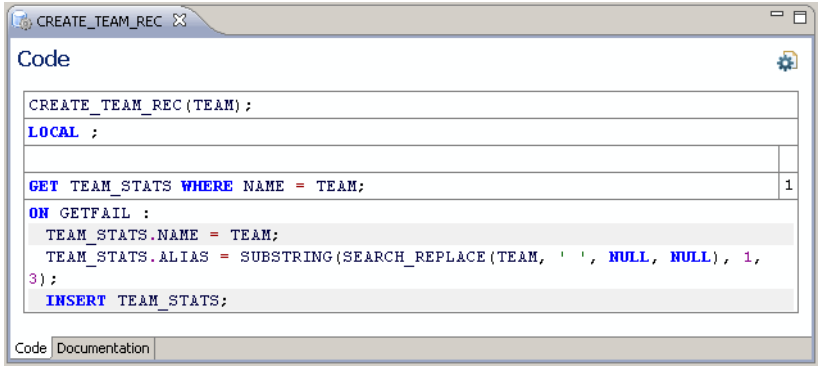
Processing the Data

The processing starts by building all the rows of the output table, one row per team, based on information from the table GAME_SCHEDULE.

The rule CREATE_TEAM_LIST loops through the table GAME_SCHEDULE and calls the rule CREATE_TEAM_REC for every HOME_TEAM and every VISITING_TEAM from that table:



Note that running this rule does not create multiple rows per team. This rule works in conjunction with the rule CREATE_TEAM_REC which is designed to add a new row to the output table only if the row for the team did not exist in the table before:



The INSERT operation only takes place if the GET operation on the row where the NAME equals to the team name fails, therefore only the first time the processing comes across that team in the TEAM_STATS table as either a visiting or home team.

Using TIBCO Object Service Broker Shareable Tools

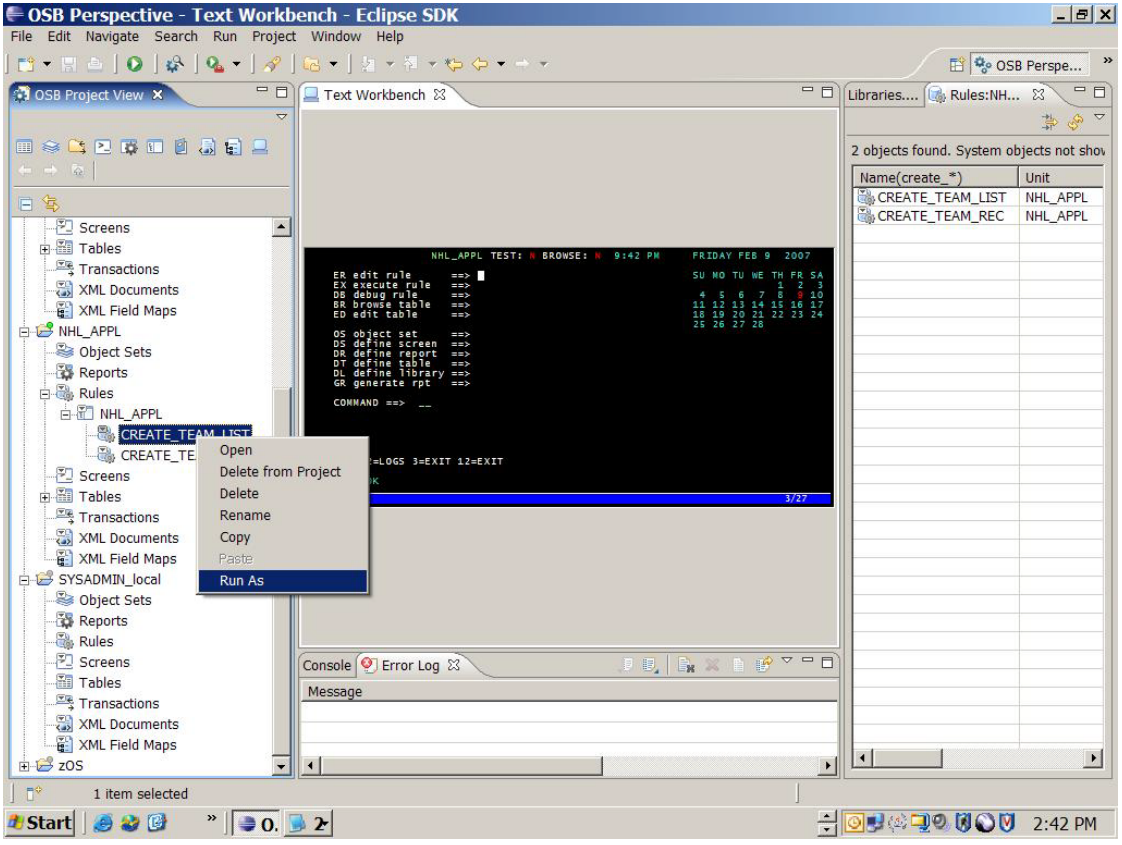
The field ALIAS of the TEAM_STATS table contains the first three characters of the team name. This is to handle the fact that the team names in the table GAME_RESULT are presented in a format different than the format of the team name in the GAME_SCHEDULE table. This situation is common when an application is accessing information from different external sources.

To devise the value of the field ALIAS, we use the shareable tools SUBSTRING and SEARCH_REPLACE. Shareable tools are programs supplied with TIBCO Object Service Broker that expedite rules language programming and application development by performing common functions and facilitating tasks such as string manipulation, mathematical calculation, and object handling. For more information about these and the other sharable tools, refer to the *TIBCO Object Service Broker Shareable Tools* manual.

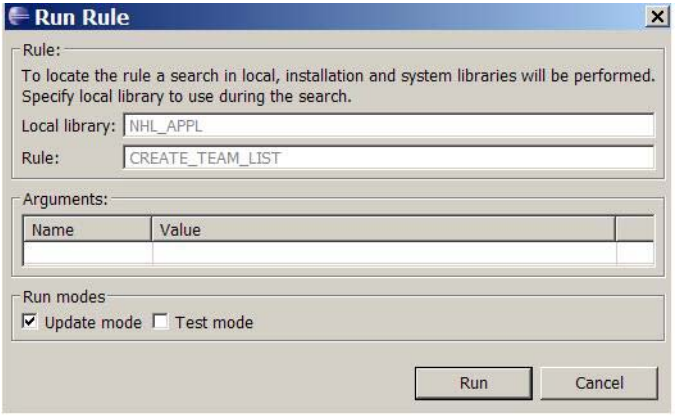
Testing your Rule

You can use the graphical or text interface to test your rule from the TIBCO Object Service Broker UI:

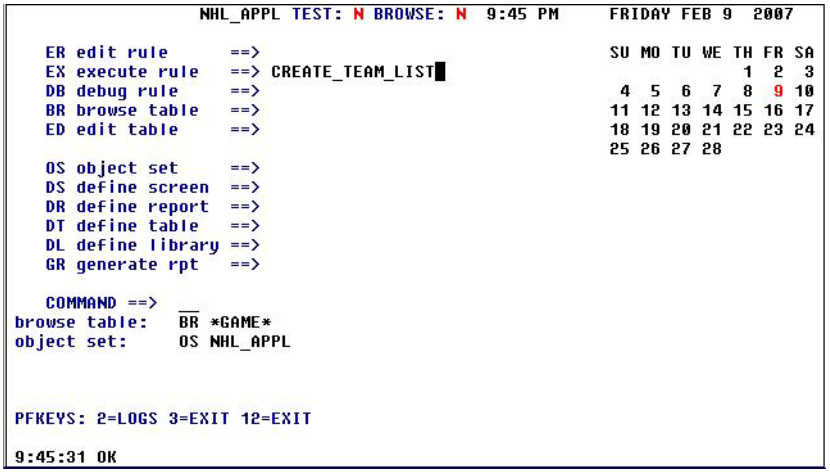
Using the UI select the rule you want to run in either the Rules Explorer view, or in the OSB Project view. Right-clicking on the name offers the menu option “Run As …”, as shown below:



You are prompted for arguments you want to pass to the rule:



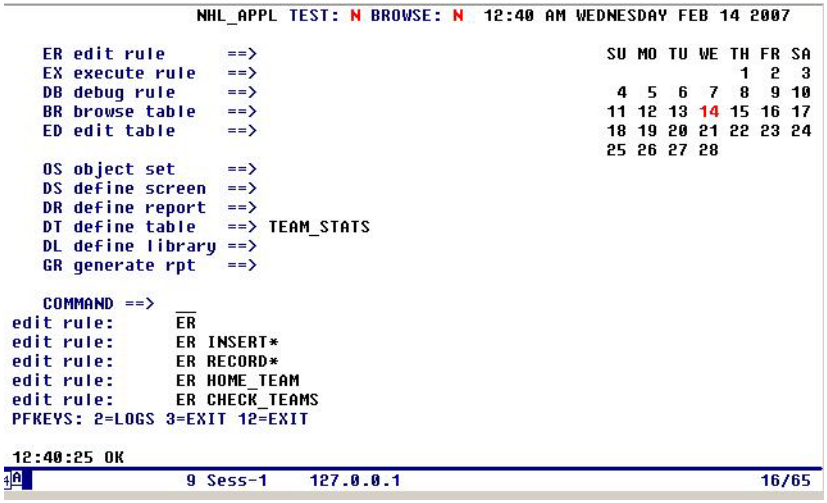
As the rule CREATE_TEAM_LIST does not take any arguments, you only need to press the Run button to start the execution of the rule. Alternatively, you can use the Text Workbench ER execute rule menu item, type in the name of the rule and press Enter:



Verifying What is Stored in a TEM Table

The result of running the rule CREATE_TEAM_LIST is a set of rows each describing one team. The table, however, is defined as TEM type table, which means that when the execution of the rule is over, the content of the table is discarded. If you try to browse or edit the table TEAM_STATS after running the rule CREATE_TEAM_LIST, the Table Editor shows an empty table.

While it is possible to use the MSGLOG shareable tool to print the rows of the table to the Message Log, formatting the print line to display the content of all fields can be a tedious task. Alternatively, you can temporarily switch the table type from TEM to SES while testing the rule from the Text Workbench. To change the table type, simply open the Table Definer Tool from the Text Workbench by typing the table name beside the DR define table menu item:



The text version of the table definition for TEAM_STATS is displayed:

```
  COMMAND==>                          TABLE DEFINITION

Table: TEAM_STATS       Type: TEM   Unit: NHL_APPL        IDgen: N

    Parameter Name  Typ Syn Len Dec Class             ,      Event Rule   Typ Acc
   ----------------  -   -  --- --   -                ,   ---------------- -   -
 _                                                    ,
 _                                                    ,
      Field Name    Typ Syn Len Dec Key Ord Rqd    Default          Reference
   ---------------- -   -  ---- --  -   -   - ---------------- ----------------
 _ NAME              S   V    15  0  P
 _ ALIAS             I   C     3  0
 _ WINS              C   C     3  0
 _ LOSSES            C   C     3  0
 _ SCORED            C   C     4  0
 _ AGAINST           C   C     4  0
```

Type the string SES instead of TEM for the field Type, and save the definition by pressing PF3 key.

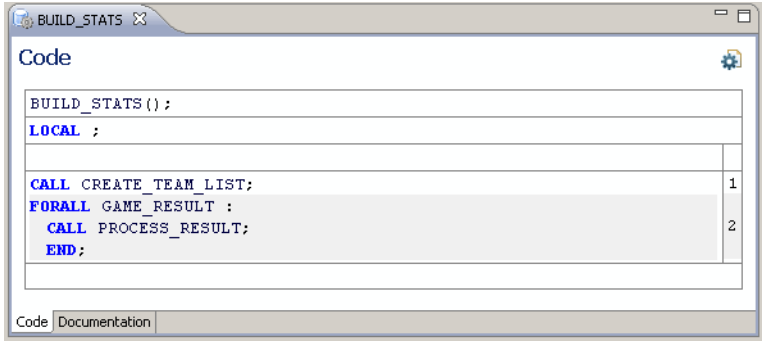
As a result the content of the table are not discarded when the transaction is finished, and it remains available for browsing inside of the same Text Workbench session:

```
BROWSING TABLE    :  TEAM_STATS
COMMAND ==>

          NAME      ALIAS WINS LOSSES SCORED AGAINST
_ ---------------- ---   ---  ---    ----   ----
_  ANAHEIM         ANA
_  ATLANTA         ATL
_  BOSTON          BOS
_  BUFFALO         BUF
_  CALGARY         CAL
_  CAROLINA        CAR
_  CHICAGO         CHI
```

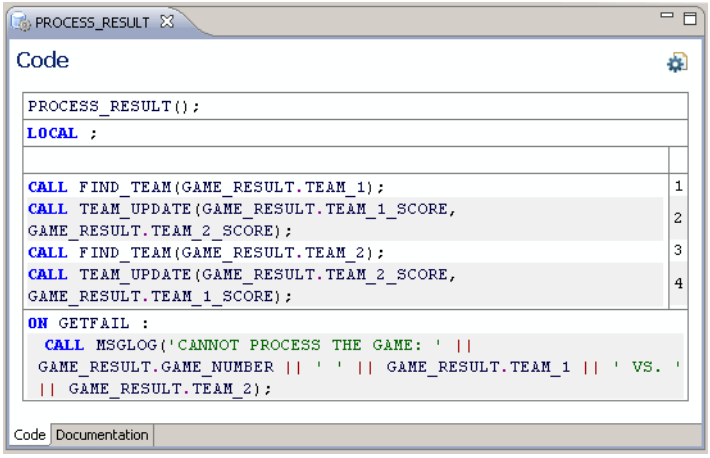
If you open a new Text Workbench inside the TIBCO Object Service Broker UI a new session is created and the table TEAM_STATS is empty. If you want to change your rule and test it again, since this is a session table, you can recycle the Text Workbench to start with an empty table.

Rules to Build Game Results

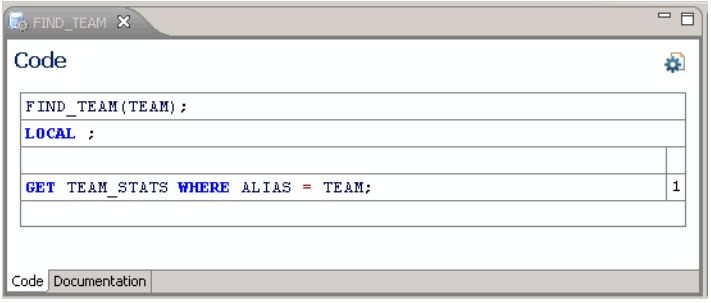
The functionality to build game results based on requirements can be accomplished in many different ways. The solution presented is just one of many possibilities:



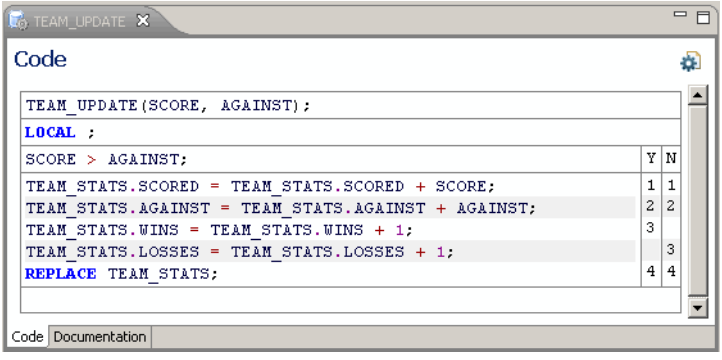
The initial rule BUILD_STATS calls the rule CREATE_TEAM_LIST described earlier. This sets up the team names and aliases in the TEM table TEAM_STATS. It then loops through the IMP table containing game results and processes each result:



Note that every row in GAME_RESULT contains the information about both contending teams. For each game result, the rule PROCESS_RESULT uses the rule FIND_TEAM twice, once for each team that played the game to retrieve its record from the table TEAM_STATS.



For each of the teams, PROCESS_RESULT then calls the rule TEAM_UPDATE, passing the number of goals the team scored and received as arguments.

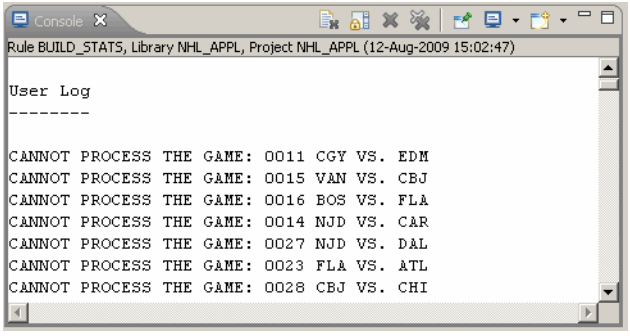


The total of goals scored by and against for the team is increased by the values passed as arguments and the number of wins or losses is updated depending on the condition statement at the top of the rule. Depending on the game results the number of wins or losses, is incremented.

After running rule BUILD_STATS, you can verify the messages produced. If run from the Text Workbench, press the PF2 key to display any messages within the Message Log:

```
-------------------------- INFORMATIONAL MESSAGE LOG --------------------------
COMMAND ===>                                                  SCROLL ===> P
CANNOT PROCESS THE GAME: 0011 CGY VS.EDM
CANNOT PROCESS THE GAME: 0015 VAN VS.CBJ
CANNOT PROCESS THE GAME: 0016 BOS VS.FLA
CANNOT PROCESS THE GAME: 0014 NJD VS.CAR
CANNOT PROCESS THE GAME: 0027 NJD VS.DAL
CANNOT PROCESS THE GAME: 0023 FLA VS.ATL
CANNOT PROCESS THE GAME: 0028 CBJ VS.CHI
```

If run from the TIBCO Object Service Broker UI, any messages will be shown in the Console Log displayed:



These messages were issued by the GETFAIL exception handler in rule PROCESS_RESULT.

If you kept the table TEAM_STATS type as SES for testing purposes, you can verify the content of the table, and notice that for some teams there are no games recorded:

```
BROWSING TABLE    :  TEAM_STATS
COMMAND ==>

          NAME      ALIAS WINS LOSSES SCORED AGAINST
_ ---------------- ---   ---  ---    ----   ----
_  ANAHEIM         ANA   27   19     151    120
_  ATLANTA         ATL   23   23     138    146
_  BOSTON          BOS   20   20     120    141
_  BUFFALO         BUF   31   16     178    138
_  CALGARY         CAL
_  CAROLINA        CAR   23   24     143    145
_  CHICAGO         CHI   16   29     110    140
_  COLORADO        COL   24   21     145    126
_  COLUMBUS        CBJ
_  DALLAS          DAL   27   18     121    110
_  DETROIT         DET   27   19     137    115
_  EDMONTON        EDM   20   23     122    127
_  FLORIDA         FLO
_  LOS ANGELES     LOS   13   33     126    174
_  MINNESOTA       MIN   22   24     124    123
_  MONTREAL        MON   25   20     139    131
_  NASHVILLE       NAS   30   15     158    120
```

Closer inspection of the text file with game results verifies that the aliases used for those teams, built as first three characters of the team name, do not correspond to the team acronyms in the result file.

This situation roughly corresponds to the situation where the external data source contains invalid or inconsistent data. The rules language allows us to add functionality to record the instances of inconsistent information or even fix it, with minimal changes to the existing processing.

Task E: Using Exceptions to Handle Data Errors or Inconsistent External Data

In this task you will write an exception handler to handle the situation where the team acronym from the external data source does not correspond to the team alias. Use the GAME_SCHEDULE table to confirm and finalize the update if the record in the external data source is otherwise valid.

Requirements to Update a Record

Let us establish what we consider enough information about the game to update a team's record, even if the team name in the result file does not matching the alias expected.

- Game number in the record from the result file has to have a matching record, with the same game number, in the game schedule table created in , Task B: Create a Table Definition to Store Data in Native TDS Table Format, page 44
- The information about the opposing team has to match up with the record from the result file and the record from the game schedule table.

Acceptable Match

As an example, records:

```
20061005 0011 cgy 1 edm 3 final

  11 Thu Oct  5  CALGARY           AT  EDMONTON        08:00 P
```

are considered to be an acceptable match because the game number is the same for both, and the acronym in the result file "edm" for EDMONTON matches the alias used for EDMONTON in the TEAM_STATS row. Therefore, the result with acronym 'cgy' will be applied to the TEAM_STATS row for CALGARY.

Unacceptable Match

On the other hand, records:

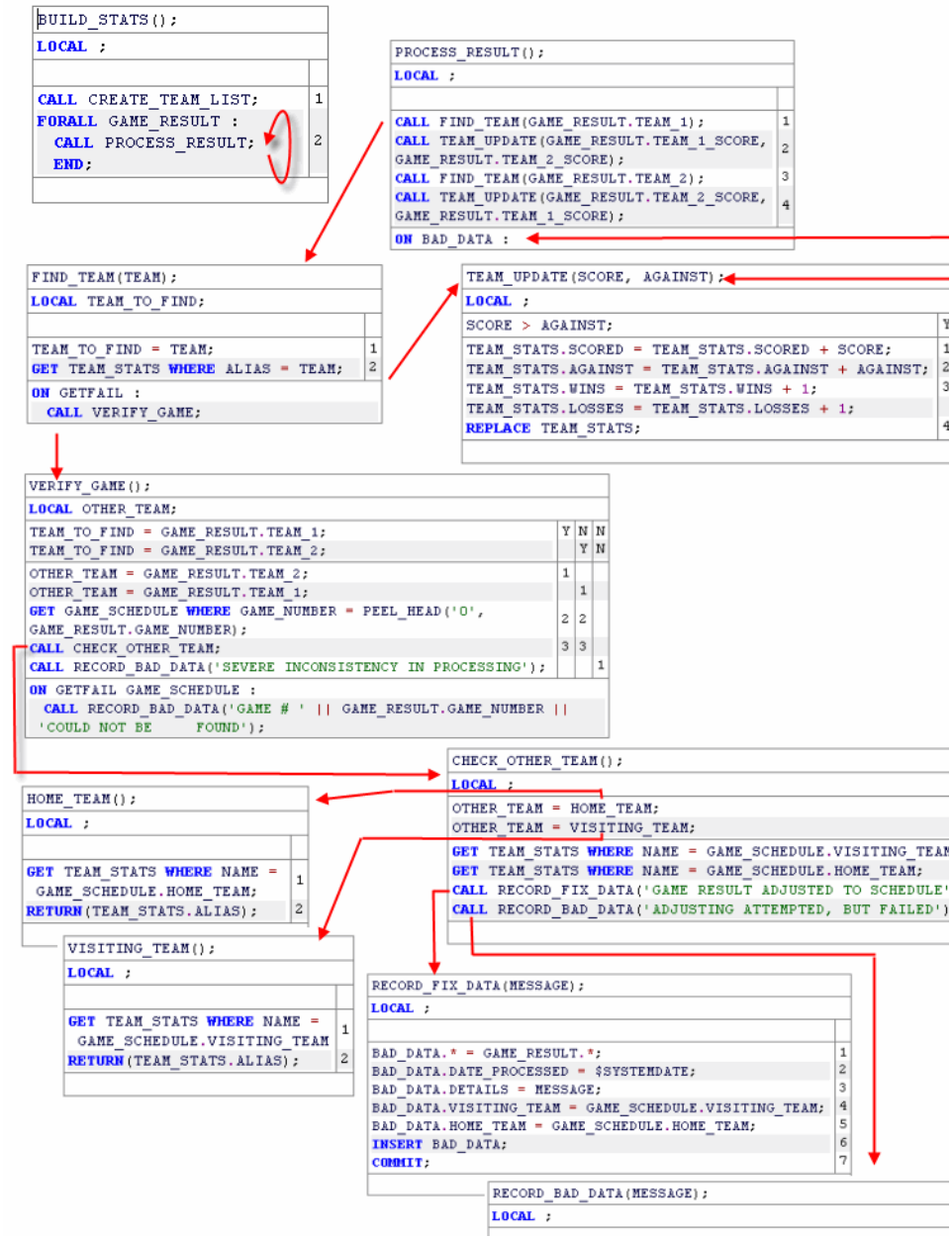
```
20061026 0138 fla 0 njd 2 final

138 Thu Oct 26  FLORIDA          AT  NEW JERSEY      07:30 P
```

are not considered as an acceptable match, because even with the same game number, neither team's alias corresponds to the value used in the result file.

Logic Flow

Well-written TIBCO Object Service Broker rules are short programs that accomplish a specific business logic operation. Each rule encapsulates all implementation details.

The following picture outlines the execution flow through the set of rules implementing , Task E: Using Exceptions to Handle Data Errors or Inconsistent External Data, page 59.



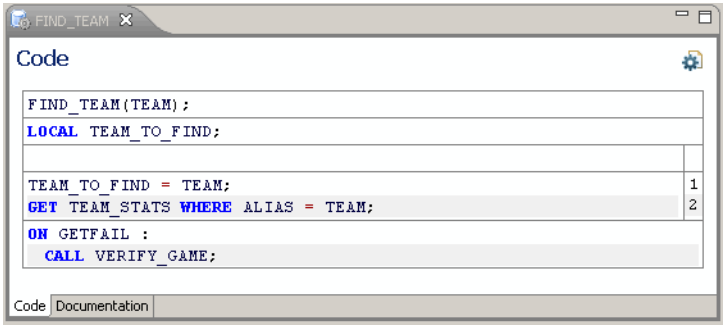
Implementation Details of the Rules

The following section outlines implementation details for each of the rules in the solution:

FIND_TEAM

We need only minimal changes to the existing code: instead of printing Message Log information when the GETFAIL exception is signaled, the custom exception handling rule VERIFY_GAME is invoked. This only happens if the rule FIND_TEAM cannot locate the correct row of the table TEAM_STATS as per the GAME_RESULT alias; if a GETFAIL exception is raised, the custom exception handler VERIFY_GAME is called.

To be able to verify if the other team in the result record provides us with the acceptable match, we have to carry information pointing to the team that could not be located in the TEAM_STATS table using the rule FIND_TEAM. We update FIND_TEAM to set the value of the local variable TEAM_TO_FIND before attempting to get the row. If the exception handler VERIFY_GAME is invoked, the local variable TEAM_TO_FIND is set and available.

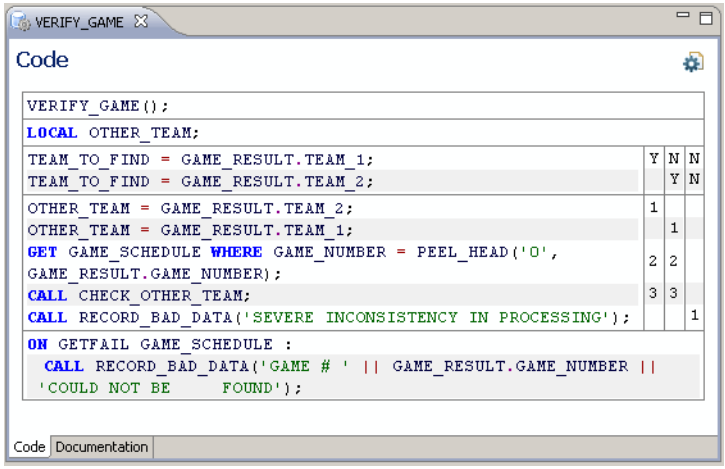


As the VERIFY_GAME rule handles the exception generated in the FIND_TEAM rule, the control returns to the caller of FIND_TEAM, the rule PROCESS_RESULT. If the rule VERIFY_GAME successfully locates the row from TEAM_STATS that should be updated (the GAME_SCHEDULE is checked, an acceptable match found, and the team found by referencing the team full name from GAME_SCHEDULE), this row will be updated by the UPDATE_TEAM rule that is invoked after FIND_TEAM.

Alternatively, if VERIFY_GAME cannot locate the acceptable match, the UPDATE_TEAM will not be called.

VERIFY_GAME

The exception handler identifies the other team from the GAME_RESULT row using conditional processing at the beginning, and saves that information as the local variable OTHER_TEAM.



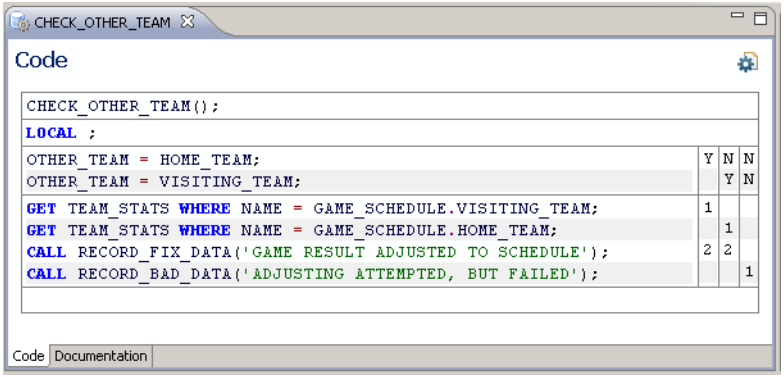
The game number from the rule GAME_RESULT is used to retrieve the game schedule from the GAME_INFO table. The rule CHECK_TEAMS is then invoked to verify if the records are a match as defined according to the requirements.

Note that this rule provides error handling for a situation where:

- The local variable TEAM_TO_FIND is not equal to either team alias from the GAME_RESULT row. This indicates an error in processing logic.
- The game with the game number as per the GAME_RESULT row does not exist in the schedule recorded in the GAME_SCHEDULE table. This indicates an inconsistency between external data sources.

CHECK_OTHER_TEAM

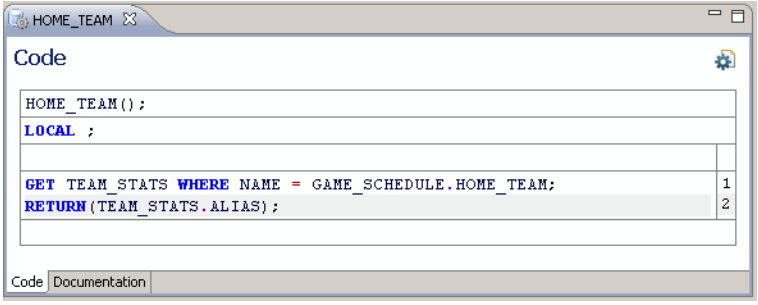
This rule verifies that the other team from the GAME_RESULT row corresponds to the team from the schedule found in the GAME_SCHEDULE table.

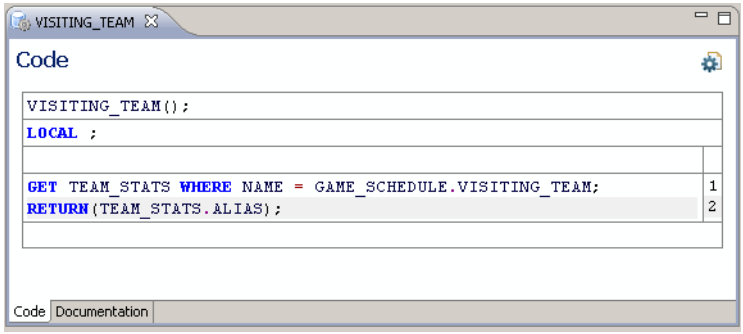


If either the HOME_TEAM or VISITING_TEAM field from the GAME_SCHEDULE table corresponds to the local variable OTHER_TEAM, we now know that the team we could not identify is the other one.

To locate the correct TEAM_STATS record to update, we now use the full team name of the “other team”. Note the WHERE clause on the GET statement.

Also, note that conditional section at the top of CHECK_TEAM uses two short functions (rules that return a value) to return both HOME_TEAM and VISITING_TEAM aliases based on the team name from the GAME_SCHEDULE table:

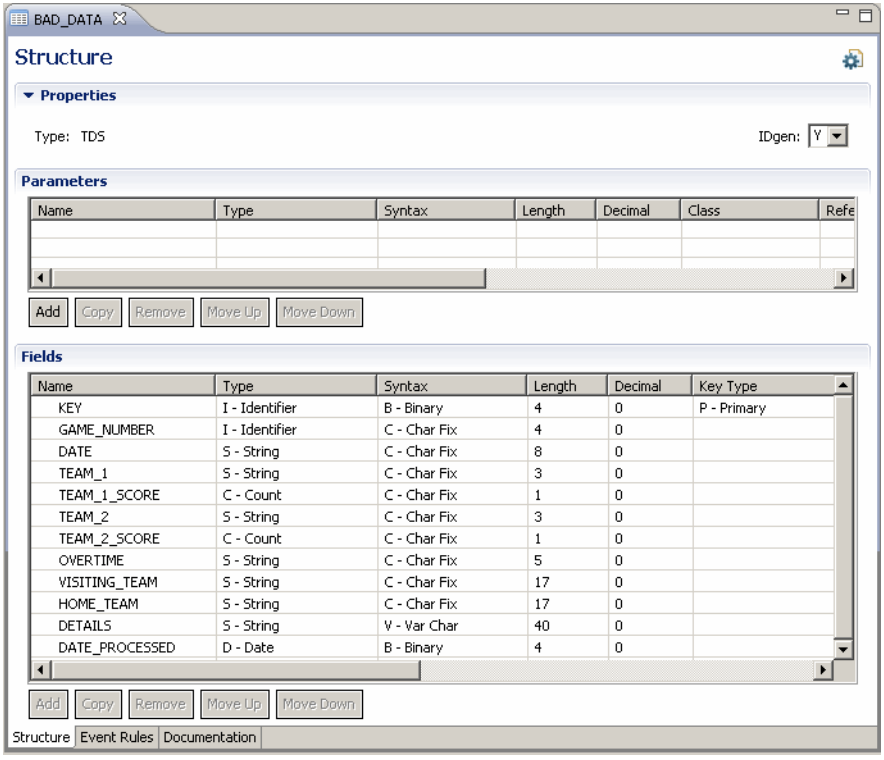




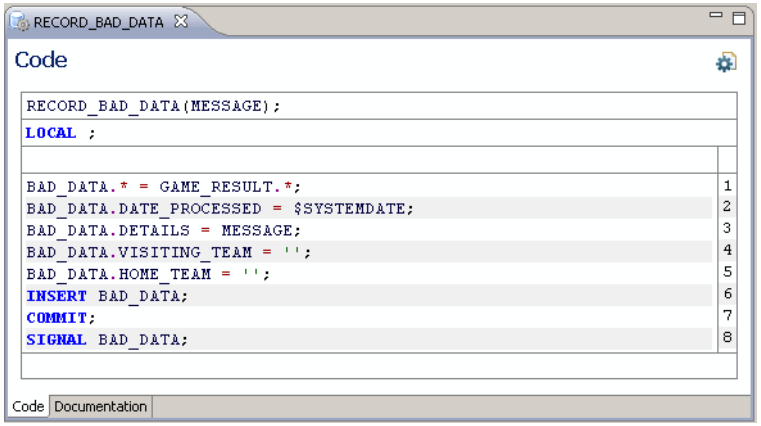
RECORD_BAD_DATA and RECORD_FIX_DATA

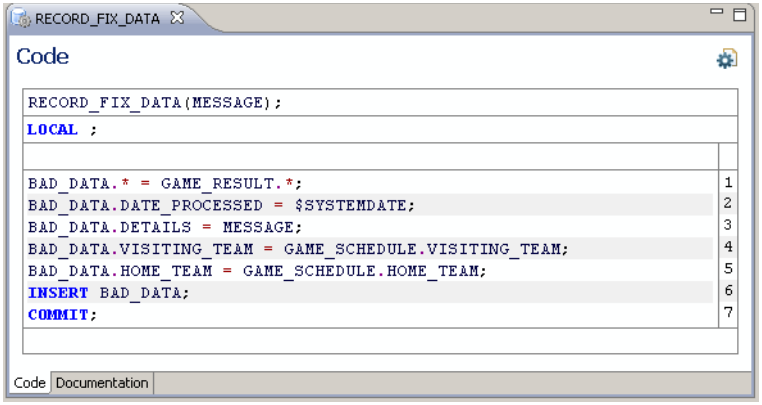
These two rules are used to keep track of the external data source information that is found to be inconsistent with the reference table or another external data source, and of the action that the application has taken to fix it if possible.

Both rules insert a row in the native TDS table BAD_DATA defined as:



This table hosts all the information from the external data source that was considered inconsistent, the details about inconsistency provided at the time processing was done, and the date of processing. It is defined to be an IDGen table, meaning that TIBCO Object Service Broker will generate a unique key for each row inserted in the table.



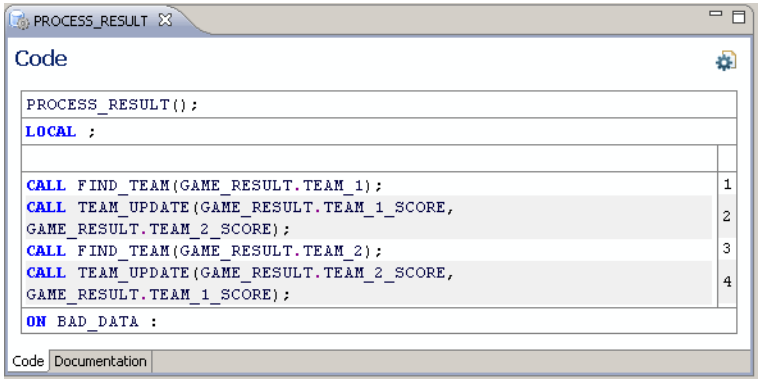


These two rules are almost identical. The significant difference is statement number 8 in the RECORD_BAD_DATA rule. This statement signals the custom exception BAD_DATA which is designed to prevent the update of the TEAM_STATS table if the correct row could not be located.

If the GAME_RESULT row is accepted as having an “acceptable match” the RECORD_FIX_DATA will be invoked instead of RECORD_BAD_DATA, and the update of the TEAM_STATS will proceed with the row that was located in the VERIFY_GAME exception handler.

The exception BAD_DATA is caught inside of the rule PROCESS_RESULT and causes the UPDATE_TEAM to be skipped. There is no additional processing associated with this exception; its only purpose is to change the flow of execution to avoid UPDATE_TEAM.

The updated rule PROCESS_RESULT is coded as follows:

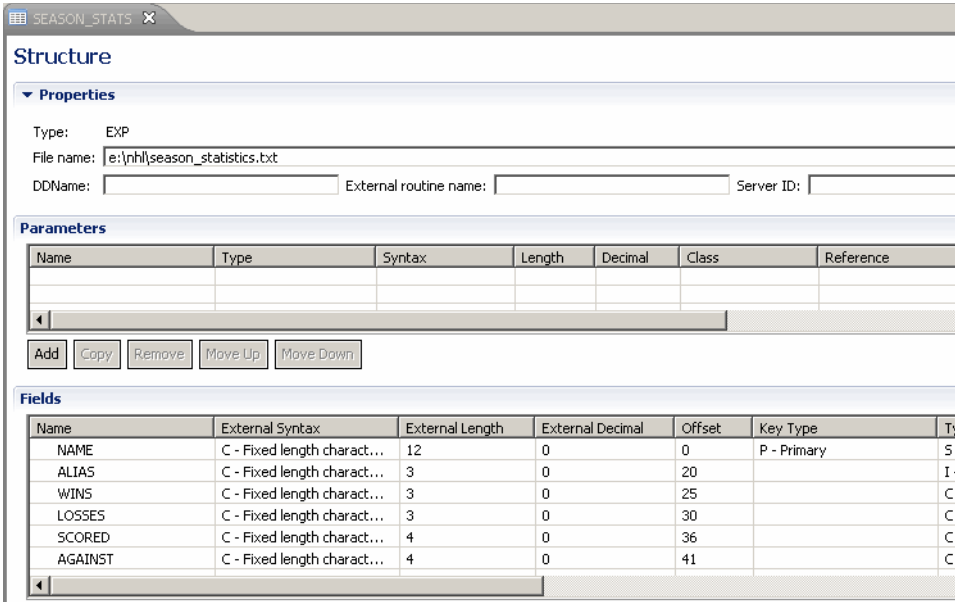


Task F: Export the Results

In this task you will create a definition for the table SEASON_STATS to map the data from the TIBCO Object Service Broker system to the external repository. It is to be used to update the external data repository

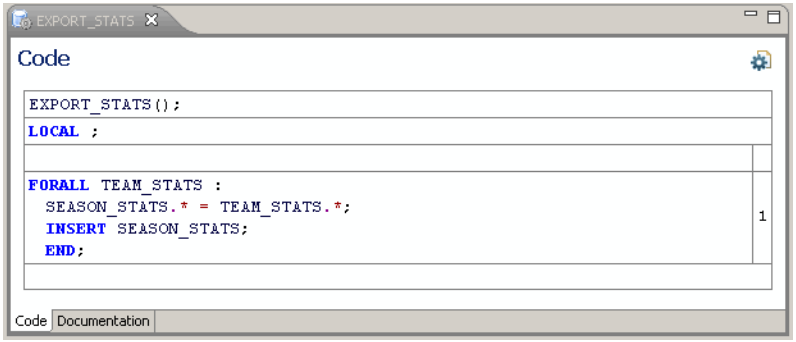
Defining an Export (EXP) Table

Using the New Table Definition Wizard and Table Definer in the same way as in Task A, we create a table of type EXP (export) to map data to the text file:



Rule to Transfer Data

To transfer data from the TEM table TEAM_STATS to the EXP table SEASON_STATS, we create the rule that for all rows in TEAM_STATS simply assigns all fields by name and inserts the row into SEASON_STATS.

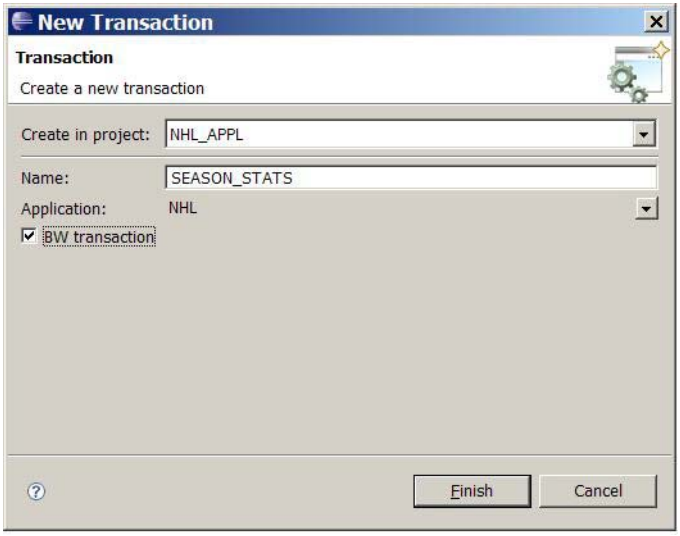


Task G: Make the Information Available to BusinessWorks

In this task you will create a Transaction object so that the information is available to TIBCO BusinessWorks. You will also make the information about inconsistent data available to the BusinessWorks process.

Defining a Transaction Object

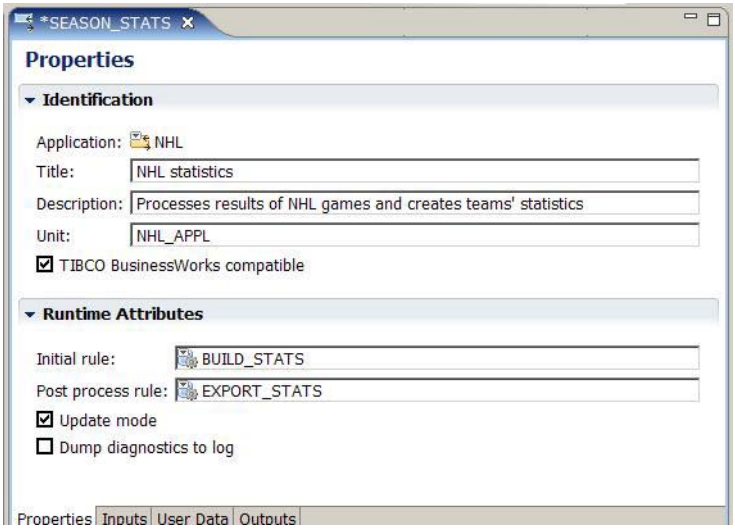
You create the TIBCO Object Service Broker Transaction object via New Transaction Wizard:



Usage Notes:

- You must select an existing application from the drop-down list.
- To create a new application, start the New Application Wizard first; you only need to specify the application name.

When you press “Finish”, the TIBCO Object Service Broker Transaction Editor Properties tab will be displayed. Enter the values as shown below:



Notes about this panel:

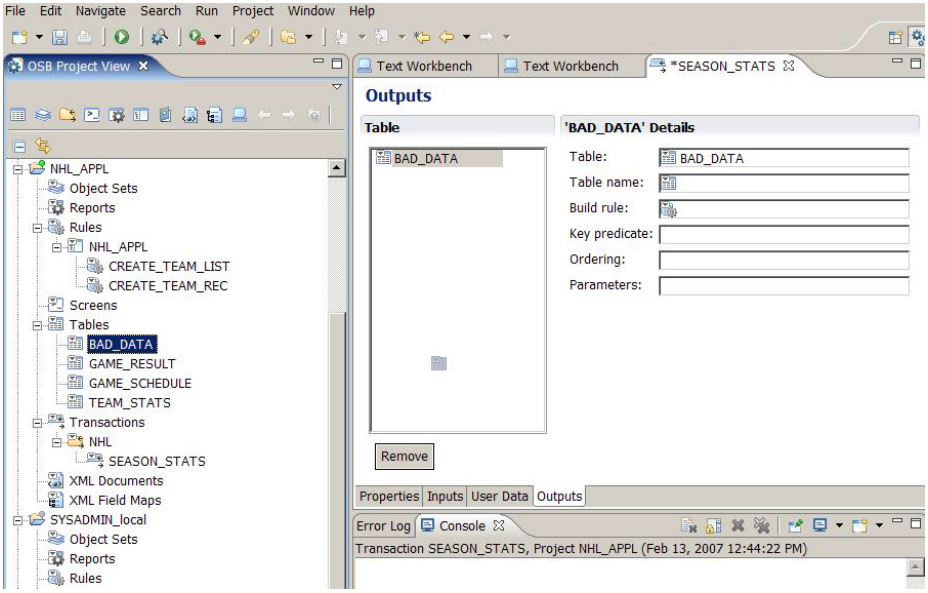
- Title and Description fields are not mandatory
- TIBCO BusinessWorks Compatible checkbox has to be checked to create a transaction that can be used in the context of BusinessWorks process
- Update mode checkbox has to be checked as the transaction updates a native TDS table.
- BUILD_STATS (created in Tasks C-E) is defined as the initial rule for this transaction
- EXPORTS_STATS (created in Task F) is defined as the post-process rule

In addition to the Properties Panel, there are three more Panels in the Transaction Editor, which are selected at the bottom of the Transaction Editor:

- Inputs: specifies Input tables used by this transaction. Only the tables that are populated with data passed by a client (that is via mapping in a BusinessWorks process) are considered Input tables. The external data sources, like the IMP table that contains the input data for our application, are not considered an Input table for the Transaction object
- User Data: specifies which User Data in the Name/Value pair format can be passed to the transaction
- Outputs: specifies tables which will be passed to the client as output from the transaction

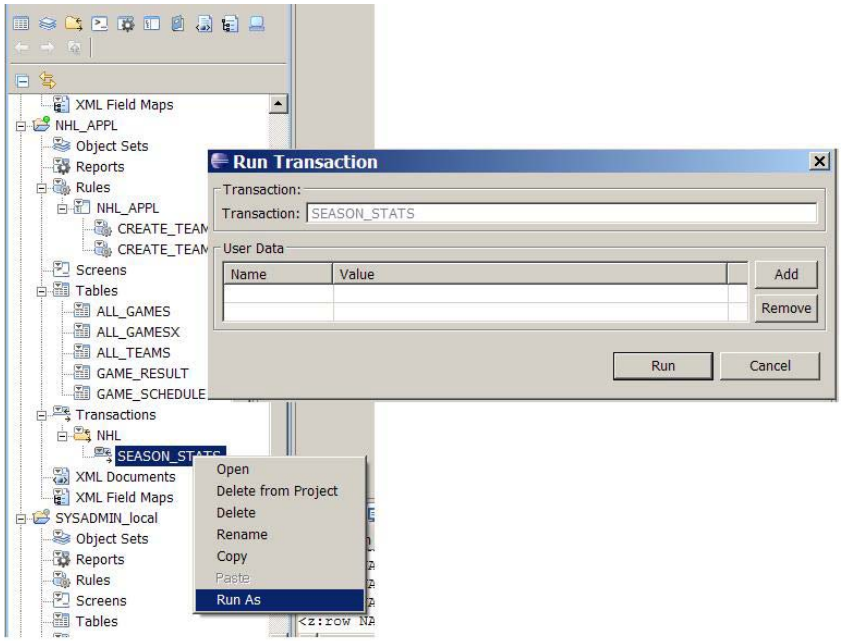
Passing the Information about Inconsistent Data to BusinessWorks

As the task requirement specifies that the information about inconsistent data has to be passed to the BusinessWorks Process, you need to drag the table BAD_DATA from the Project View to the list of Output Tables:



Testing the Transaction

You can test the transaction using the “Run As …” menu item from either Transaction Explorer or Project View:



The transaction creates the text file with each team's statistics, and the content of the output table BAD_DATA is displayed in XML format inside the Console area of the UI:

Final Steps

As the transaction is now tested and ready for use, it is important to change the table type for TEAM_STATS back to type TEM. This table is designed to hold temporary information only kept for the duration of the database transaction and we switched its type to SES (kept for the life span of the session) so that we could inspect its content during the development cycle.

In production, the assumption is that we always start the transaction with no data in the TEAM_STATS table, and if it was left as SES type, it would accumulate the information from every invocation within the same session.

Cleaning Up Session Tables

Session (SES) tables are used in TIBCO Object Service Broker applications mainly to transfer data across the database transaction boundaries. It is extremely important to use SES tables with caution and delete their content immediately when it is not required by the business logic.

Appendix A Common TIBCO Object Service Broker Text-Based Workbench Functionality

This appendix describes how to use common function keys and primary commands available for the text-based workbench tools, navigate the MetaStor, and document TIBCO Object Service Broker objects.

TIBCO Object Service Broker also provides a graphical environment for TIBCO Object Service Broker development. For information about using the TIBCO Object Service Broker UI, refer to the TIBCO Object Service Broker UI online help.

Topics

Common TIBCO Object Service Broker PF Keys

The PF key line at the bottom of a screen displays selected PF keys for each screen. Due to space limitations, not all the available PF keys appear at the bottom of some screens. To view all the available PF keys for a screen, press PF1 in an area other than a user field, because user fields can have their own field-level help.

In your applications, you can reprogram most of the standard PF keys. Non-standard PF keys and variations of these keys are discussed in the appropriate manual for that tool.

Common PF Keys

The following PF keys are common in TIBCO Object Service Broker:

PF1	Display help.
PF2	Display a documentation screen, or the message logs.
PF3	Exit the screen or tool.
PF7	Scroll vertically up.
PF8	Scroll vertically down.
PF9	Recall last command.
PF10	Scroll horizontally right.
PF11	Scroll horizontally left.
PF12	Exit the screen or tool.
PF13	Print an object.
PF22	Delete a definition.
PF24	Refresh the screen.

For terminals that do not support the full set of PF keys, press Ctrl+F followed by the two-digit number for the function key you want to use. For example, to use PF3, press CTRL+F, 0, 3.

Display Help–PF1

To invoke the Help facility, press PF1. There are three levels of help available: field, screen, and general help.

Display a Documentation Screen–PF2

To display a Documentation screen for rules, screen, table, and report objects stored in the MetaStor, press PF2. The Documentation screen provides a summary and description of the object, and tells you when it was created and last modified. You can do a keyword search using values provided in this screen.

Refer to Object Documentation on page 91 for details.

Display Message Logs–PF2

To display message logs from the last tool run from the workbench, press PF2.

Refer to Messaging Mechanisms on page 33 for more information.

Exit the Screen and/or Tool–PF3/PF12

When exiting the screen and/or tool, you can do one of the following:

- To save any changes you made and either end your current session or return to the previous screen, press PF3.
- To cancel the changes and either end your current session or return to the previous screen, press PF12.

Scroll Vertically–PF7/PF8

When available, use PF7 to scroll up and PF8 to scroll down the screen. To modify the default scrolling amount of one screen, do the following:

1. Type a valid character in the primary command field, if present, or in the scroll field.

   For a list of valid scroll characters, press PF1.

2. Press either PF7 or PF8.

Scroll Horizontally–PF10/PF11

Use PF10 to scroll to the left and PF11 to scroll to the right of the screen. To modify the default scrolling amount of one screen, do the following:

1. Type a valid character into the primary command or the scroll field.

   For a list of valid scroll characters press PF1.

2. Press either PF10 or PF11.

Recall the Last Command–PF9

Use PF9 to recall the last primary command. You can recall up to 10 commands.

Print an Object–PF13

Use PF13 to print any object (rule, screen, table, or report) you are viewing. A message at the bottom of the screen confirms the execution of the command.

Delete a Definition–PF22

Use PF22 to delete an object's definition (table, screen table, and so on). You are prompted to confirm the deletion.

Refresh the Screen–PF24

Use PF24 to refresh your screen to the state it was in after the last command issued (works for primary and line commands).

This key cannot be reprogrammed.

Primary Commands

Where a command line is present, primary commands are word interpretations of commands issued by PF keys. The commands perform various functions on the entire screen, depending on which screen you are using. Primary commands provide the following benefits over PF keys:

- Bypass of keyboard limitations
- More explicit commands such as a selection
- Retrieval of a previously issued command

You can use the shortest unique truncation of the primary command when issuing it. For example, **DE**, **DEL**, or **DELE** are all valid forms of **DELETE**.

Example

The following example illustrates the **primary command** field, on a screen other than the main menu screen, with the **CANCEL** primary command typed in the field:

```
Command ==> cancel
```

Using Primary Commands

To use primary commands, complete the following steps:

1. Type the command name in the primary command field.
2. Press Enter.

Standard Primary Commands

The following is a list of standard primary commands. Non-standard primary commands are discussed in the appropriate manual for the tool you are using.

Primary Command	PF Key	Description
CANCEL	12	Cancels changes made and exits from the existing screen.
DELETE	22	Deletes an occurrence or object. You are prompted to confirm the deletion. If you decide to cancel the deletion, press any PF key other than the corresponding Delete PF key. When the deletion is complete, the updated screen appears.
DOCUMENTATION	2	Invokes the Documentation screen for the object you are viewing.
PRINT	13	Prints the definition or data that you are viewing. The output is routed to a predefined destination on z/OS. On Open Systems, the destination of printed output is determined by parameter settings.
SAVE	3	Saves the changes made. Save could also cause you to exit from the existing session or screen.

The Navigation Tool

Object Manager

From the workbench, the Object Manager tool provides navigation of the MetaStor. The Object Manager is a TIBCO Object Service Broker program that handles lists of objects associated with a particular tool. Using the Object Manager, you can view and manipulate the objects in the MetaStor.

Example

The following example illustrates the Object Manager screen for a list of tables provided with TIBCO Object Service Broker.

```
                          List of tables to browse
 Command ==>                                                      Scroll P
            Enter one or more line commands or a primary command
         NAME        TYPE CREATOR   CREATED     UNIT              DESCRIPTION*
    ---------------- ---- -------- ---------- -------- ----------------------
  _ ABC1             TDS  USR40    2000-04-07 USR40
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR
  _ LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH
  _ DATATEST         TDS  RPSUSR   2000-03-06 RPSUSR
  _ NRGTEST          TDS  USR40    2000-03-03 USR40
  _ USRWEB           TDS  USR00    2000-02-22 USR00
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL
D-Del Occs  P-Print  S-Select
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
9:10:46 OK
```

Managing the Selection

To modify the list of objects on the Object Manager screen, for example by selecting only certain members or by changing the order of presentation, or to act upon a subset of it, you can use the following four primary commands:

- **FIND**
- **SELECT**
- **ORDERED**
- **APPLY**

You can use operators and wildcard characters to modify these commands. Refer to Available Operators on page 86 for more information.t

If a table has parameters, you are prompted for them in the same manner as from the workbench.

Using Wildcards in the Workbench

From the workbench, use these wildcards to pre-select an object manager object list:

*	A string of zero or more characters.
?	A single character.

Example

The following example displays the Rule Editor object manager screen with a listing of only the rules that begin with ABC:

```
ER edit rule ===> ABC*<ENTER>
```

Recalling Previous Commands

A history of ten previous commands is available from the Object Manager screens of the TIBCO Object Service Broker workbench tools. Press PF9 up to a maximum of ten times until you reach the previous primary command that you require. This limit is available for each tool and lasts for the duration of your TIBCO Object Service Broker session.

FIND Command

The **FIND** command locates an occurrence that meets the selection criteria you specify The search starts at the cursor position, not at the beginning of the selection. If an occurrence that meets the condition is found, the cursor is placed to the left of its primary key, which is highlighted. To find a subsequent occurrence that meets the same criteria, press PF5 (**FIND NEXT**).

The following restrictions apply for the **FIND** command:

- All non-numeric constants must be enclosed in single quotation marks.
- The maximum length of a value is the length of the primary command line.
- The **FIND** command on the **DESCRIPTION** field is case sensitive.

Examples of the FIND Command

- Type `find salary > 700` to locate the first occurrence of an employee whose salary is greater than $700.
- Type `find salary > 700 & region = 'midwest'` to locate the first occurrence of an employee whose salary is greater than $700 and whose region is the midwest.

SELECT Command

The **SELECT** command locates and displays a subset of occurrences of a table that meet the selection criteria. After you issue a **SELECT**, you can view all the occurrences in the table again by issuing the **SELECT** command with no selection criteria.

- All non-numeric constants must be enclosed in single quotation marks.
- The **SELECT** command on the **DESCRIPTION** field is case sensitive.
- The **SELECT** command is generally faster than the **FIND** command.
- When using the **SELECT** command, a subset of the original occurrences appears. Subsequent commands pertain to this subset, except if you issue another **SELECT**, **ORDERED**, **EDIT**, or **BROWSE** command, which functions on the whole table, not just the subset. After issuing a **SELECT**, you can view all the occurrences in the table again by issuing the **SELECT** command with no selection criteria.

Examples of the SELECT Command

- Type `select lname='smith'` to display all the employees with the last name Smith.

- Type `select lname='smith' & salary < 1000` to display all the employees with the last name Smith whose salary is less than $1000.

ORDERED Command

The occurrences in a table are stored in ascending order by primary key. However, you can display the information in descending order. If you want to order by a certain field, type the **ORDERED** command and then the field name on the command line. You can also specify multiple levels of ordering in one command. When specifying ascending or descending order, you can use the short forms, Asc or Desc.

Ordering data requires extensive system resources when large tables are involved.

Examples of the ORDERED Command

- Type `ordered desc deptno` to sort the occurrences of the table in descending order by department number.
- Type `ordered desc deptno and ordered asc lname` to sort occurrences in the table in descending order by department number and ascending order by last name.

Combining SELECT and ORDERED

You can combine the **SELECT** and **ORDERED** commands to display selected occurrences in a specified order. The results of a **SELECT...ORDERED** command are not stored in memory. A subsequent **SELECT...ORDERED** command affects the entire set of occurrences, not just the occurrences included in the most recent **SELECT...ORDERED**.

Example

Type `select deptno=10 ordered lname` to select all the employees in department 10, ordered according to last name.

ORDERED must be abbreviated to no less than **ORD** when used in a context such as in this example, to avoid confusion with the **OR** operator (**OR**).

APPLY Command

You can use APPLY to execute a line command for a set of occurrences using **APPLY**. The executed line command is applied to the selected items when you press Enter. You can use this command in conjunction with WHERE and **ORDERED**. For more information on line commands, refer to Managing MetaStor Objects on page 89.

- If you do not use the **APPLY** command on a selected subset, the **APPLY** command acts on all the occurrences.
- If you use the WHERE clause with **APPLY**, the selected subset appears.
- If you use the **APPLY** command after you select a set of occurrences, it acts only on this subset of occurrences.

Example

Type `apply p where creator='USR40' ordered desc name` to print all the tables created by user ID USR40 in descending order by name.

See Also *TIBCO Object Service Broker Programming in Rules* for information on the WHERE clause.

Available Operators

Operators are used in conjunction with primary commands to narrow the scope of the command. The following operators are available:

- Multiple condition operators (AND, &, OR, |)
- Relational operators (=, ¬=, <, >, <=, >=)
- Partial match operators (LIKE)

Multiple Conditions with AND

You can connect several conditions in a command using the operator AND (or the symbol &). AND means that the conditions before and after the AND are both included in the search criteria.

Example

Type `find manager='simons' and deptno=10` to locate those employees whose manager is Simons and who work in department 10.

Multiple Conditions with OR

You can use the operator OR (or abbreviation |) to connect several conditions. OR means that condition one is true, condition two is true, or both conditions are true.

Example

Type `select manager='simons' or deptno=10` to select those employees, regardless of their department, whose manager is Simons, and all those employees who work in department 10.

Those employees who have Simons as a manager and are in department 10 are selected only once.

Comparisons with Relational Operators

You can compare two values using the following operators:

=	Equal.
¬=	Not equal.
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater than or equal to.

For ostty and some 3270 emulators, the not sign (¬) symbol displays as a caret (^).

Example

Here are some examples of conditions:

- Type `find deptno=30` to locate employees who are in department 30.
- Type `find position¬='sales'` to locate all employees who are not in sales.
- Type `find salary>=986.73` to locate all salaries that are greater than or equal to $986.73.

Partial Matches with LIKE

You can use **`LIKE`** to perform partial matches when using **`SELECT`** or **`FIND`**. The string that you use must be enclosed in single quotation marks.

You can use the following wildcard characters in conjunction with LIKE:

*	A string of zero or more characters.
?	A single character.

Example

The following are some examples of inquiries that use LIKE and wildcards in a **FIND** command. In these examples, only the first employee is found. To find more employees, use the Find Next PF key (PF5) or use a **SELECT** to find all employees that match the selection criteria:

- Type `find lname like 'C*'` to locate the first employee whose last name begins with C, regardless of how many characters follow the C.
- Type `find lname like '????G'` to locate the first employee whose last name is five characters long and ends in G.
- Type `find lname like '?H*LD'` to locate the first employee whose last name has H as the second character and LD as the last two characters.
- Type `find deptno=50 and lname like '?I*'` to locate the first employee of department 50 whose last name has I as the second character.

Managing MetaStor Objects

Manipulating Objects

The list of objects on the Object Manager screen can be manipulated using various line commands. Line commands are actions that must be executed such as copy, delete, select, and so on. They are used in many of the textual interface tools to enable a single command or a block of commands to be executed at the same time.

To manipulate objects using line commands, complete the following steps:

1. Type a line command(s).

   Line commands are typed in the **line command** field denoted by an underscore at the left side of the display.

2. Press Enter.

   Line commands are processed in order from the top to the bottom of the screen.

Line Commands

The Object Manager screen has a different set of valid line commands for each tool. For information on non-standard line commands, refer to the appropriate manual for the tool you are using. The following table lists the standard line commands:

Line Command	Description
`D`	Deletes the object, after prompting for confirmation.
`P`	Prints a hard copy or prints to a file, according to settings in the user profile or according to parameter settings.
`S`	Selects the object.
`I`	Inserts a new line. This creates a new empty line below the selected line and places the cursor in column one of the new line. This is the only line command that you can issue from the top **line command** field.

Line Command	Description
C	Copies a line after the line where the cursor is located. You can use this command in conjunction with the **A** and **B** line commands.
M	Moves a line to a new position. You use this command in conjunction with the **A** and **B** line commands.
A and **B**	Specifies a destination for a move or a copy. **A** (after) indicates to copy or move the line after the one where the A is placed, and **B** (before) indicates before where the B is placed.

Example

The following is an example of the line command **P** (print) used to print a table from the Object Manager screen. In this example, the **P** line command prints the contents of the table, while using PF13 would print the list of tables.

```
                          List of tables to browse
 Command ==>                                                      Scroll P
            Enter one or more line commands or a primary command
          NAME        TYPE CREATOR   CREATED     UNIT              DESCRIPTION*
   ---------------- ---- -------- ---------- -------- ------------------------
  _ ABC1             TDS  USR40    2000-04-07 USR40
  _ DATA_SRCH_EX     TDS  RPSUSR   2000-04-06 SEARCH
  _ DATA_SRCH_EXCLUD TDS  RPSUSR   2000-04-06 SEARCH
  _ FOOTA            TDS  ABC10    2000-04-03 ABC10
  _ FOOTAS           SUB  ABC10    2000-04-03 ABC10
  _ DATA_SRCH        TDS  RPSUSR   2000-03-28 SEARCH
  _ LOCTEST          TDS  RPSUSR   2000-03-09 RPSUSR
  P LOCTESTS         SUB  RPSUSR   2000-03-09 RPSUSR
  _ DOC_TABLES       TDS  RBPADM   2000-03-09 RPSUSR
  _ SESSTST2         TDS  RPSUSR   2000-03-06 RPSUSR
  _ DATA_SRCH_ERR    SES  RPSUSR   2000-03-06 SEARCH
  _ DATA_SRCH_RESULT TDS  RPSUSR   2000-03-06 SEARCH
  _ DATATEST         TDS  RPSUSR   2000-03-06 RPSUSR
  _ NRGTEST          TDS  USR40    2000-03-03 USR40
  _ USRWEB           TDS  USR00    2000-02-22 USR00
  _ @EMPLOYEE2       TDS  USR50    1999-12-14 DOCEXMPL
D-Del Occs  P-Print  S-Select
PFKEYS: 12=EXIT 13=PRINT 3=END 5=FIND NEXT 9=RECALL
```

Object Documentation

Documentation Screen

Object Service Broker provides an interface to document your application objects. Providing and maintaining documentation for all objects encourages better management of objects. You use the Documentation screen to create and modify documentation for a particular object.

Example

This example illustrates the Documentation screen for the screen SAVE_EXPENSES:

```
DESCRIPTION OF SCREEN           SAVE_EXPENSES               UNIT: USR40

MODIFIED ON             BY                CREATED ON 23 APR 2000 BY USR40

KEYWORDS: EMPLOYEE EXPENSE
SUMMARY : TOTAL EXPENSE BY EMPLOYEE BY MONTH

                              DESCRIPTION
_ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_ THIS SCREEN LISTS THE EXPENSES FOR EACH EMPLOYEE ON A MONTHLY BASIS.




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

Invoking the Documentation Screen

To invoke the Documentation screen, complete the following steps:

1. Access the definition of the object through the appropriate definer.
2. Press PF2 or type doc in the primary command field.

   The Documentation screen appears.

The **Description** field appears formatted in browse mode if information already exists in it. If you have modify-definition access to the object, you then have the ability to modify the contents of the screen. To toggle between edit and browse modes, use PF5.

Maintaining Object Documentation

See Also

- **The SEARCH tool in the *TIBCO Object Service Broker Shareable Tools* manual for information on the Keyword Search facility**
- **The SCRIPT tool in the *TIBCO Object Service Broker Shareable Tools* manual for a list of Script commands**

Maintaining the Fields

Tools like the Table Definer or the Screen Definer update some of the fields on the Documentation screen automatically but you must maintain the following fields:

- **KEYWORDS**
- **SUMMARY**
- **DESCRIPTION**

KEYWORDS Field

Enter individual words that briefly describe the object. These words are used by the Keyword Search facility in TIBCO Object Service Broker. This field is one line long and can contain multiple entries separated by commas or blanks. For example, EMPLOYEE EXPENSE is treated as two keywords.

SUMMARY Field

Enter a one line summary of the **DESCRIPTION** field. For example, `Total Expense By Employee By Month`.

DESCRIPTION Field

Enter information about the object (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 enter in this field.

Glossary

The following items are terms used within the TIBCO Object Service Broker product set.

A

accesslog

See *audit log*.

across-by field

A report field used to determine which data columns are to be printed on an across report. When a new value is encountered for an across-by field, a new data column is printed.

across report

A report that displays its data across the report in a spreadsheet-like format. See *across-by field*.

action

A statement within a rule. An action can occupy more than one line within a rule.

action sequence number

An identifier that determines which rules statements are executed when a given condition is satisfied.

Administration menu

A menu-based administrative tool used to monitor and control the TIBCO Object Service Broker operating environment. It is invoked by the S6BTLADM/hrntladm utility.

administrator workbench

A TIBCO Object Service Broker supplied session menu used by system administrators to administer the TIBCO Object Service Broker database. To invoke the administrator workbench, @ADMIN must be specified in the security user ID profile as the session menu to be used at login.

argument

A value that is passed to a rule.

argument list

An ordered collection of arguments specified within parentheses after the rule name. The argument separator is a comma.

assignment-by-name

The values of fields of one table are assigned to identically named fields of another table.

audit log

A Security facility that logs security failures, security related events such as logins, logouts, changes of permissions, accesses to tables when logging of accesses is requested, and, if specified, all updates made by level-7 users. Also known as the accesslog.

B

Backus-Naur Form

A formal notation for describing the syntax of a programming language. It is used by the documentation to describe the syntax of the rules language. Formerly known as Backus normal form.

banner page

The title page of a report. It is normally created using a break table.

batch client

A TIBCO Object Service Broker client that is executed in batch mode.

bind

The copying of definitions or data into the Execution Environment binding area when the definition or data is first accessed by a session. Definitions of screens, tables, or rules, and the data of tables can be bound. Binding improves performance by reusing information stored in the binding area thereby reducing message traffic.

BLOBs

The acronym for binary large objects, which are typically large bitmap fields.

BNF

See *Backus-Naur Form*.

BOM

See *Byte Order Mark*.

body report table

The repeating table of a report, used to display the report data. Data from data tables is inserted into the body report table.

borrower

A user who has the Borrower flag in their user profile set to Y. A borrower can obtain rights to an object so that other users cannot modify the object. See also *rights*.

break fields

The report fields used to determine where a report is to take a control break. A new break within a body report table begins when a changed value is encountered in the break field.

break table

A repeating table, with title and body components, that groups together related information in a body report table. A new grouping is started when a new value is encountered in the break field defined to the break table. Break tables can also be defined at the beginning or end of the entire report.

browse mode

A read-only method of viewing data that does not lock the data. When a user is operating in browse mode, other users have access to the same data.

builtin

See *routine*.

Byte Order Mark

The U+FEFF (ZERO WIDTH NO-BREAK SPACE) Unicode character when used at the beginning of text to indicate the byte order and Unicode transformation of the following text.

C

C

Used for shareable tools to denote that a tool is callable. See also *E* and *F*.

Call Level Interface

A set of client and server facilities that enable programs written in third generation languages to access TIBCO Object Service Broker. Also known as the CLI. See also *SDK (C/C++)* and *SDK (Java)*.

calculation table

A table that is used to count the number of occurrences of a particular value in a specified field or fields of another table. It has a table type of CLC.

casing

The act of transforming a character from uppercase to lowercase or vice versa.

category name

A short name describing the category (that is, purpose) of a change request. The name is within a list from which users must make a selection.

change request

A request made by developers to promote changes made to objects associated with an application. When the request is accepted, the objects of the change request can be promoted. Commonly abbreviated as CR.

child rights

As used by Promotions, the borrowing rights of child objects that are part of a compound object.

classification level

A security level assigned to each object when it is created. The object is assigned a classification level that matches the object creator's security clearance level. To access an object, a user's clearance level must be equal to or greater than the object's classification level.

clearance level

A security level assigned to each TIBCO Object Service Broker user ID. The Security Manager currently supports clearance levels 0, 1, and 7. Level-0 is used only to suspend a user from logging in. Level-1 clearance is assigned to users and developers. Level-7 clearance is assigned to system administration personnel.

CLI

See *Call Level Interface*.

client

A process that requests and receives services from another process.

code page

This term is used in the context of National Language Support (NLS). It refers to an ordered set of characters and/or symbols that have a numeric index encoding (code point value) associated with each character and symbol. The term is usually used with a modifier, for example, the U.S. English EBCDIC code page. It is sometimes referred to as a character set or charset.

coercion

The act of converting a selection term prior to resolution.

command history

A scrollable section at the bottom of a standard workbench that lists commands you previously entered.

command field

See *primary command field*.

commit

An action that takes place during transaction processing that causes changes made to the database to be irreversible. This action occurs as part of normal termination of a transaction, or can be explicitly initiated with the COMMIT statement.

composite primary key

Two or more fields whose values, when concatenated together, uniquely identify occurrences of a table.

condition

An expression evaluated for its truth or logical value (Y or N). The evaluation is used to determine the flow of execution in a rule.

consolidated change request

A change request that contains within it one or more change requests with a status of C (that is, promoted change requests).

control

A control is the lowest level of graphical object that a user can manipulate within a graphical application. A control must be within a container object.

control fields

Fields of a body report table that determine how data should appear on a report. There are four types of control fields: sort, break, summary, and across-by.

control access

As used by Security, an access mode that gives a user the authority to update the security permissions to a given object. Users with control access to an object can add, modify, or delete permissions, but they cannot pass control access to anyone or take control access away from anyone. Only individual users can control objects. Security groups cannot be assigned control access.

Control Region

See *Data Object Broker*.

conversational transaction

A transaction that has interactions with a client process during the life of the transaction. The transaction retains control of all its resources while it waits for user input (that is, during a screen display). In TIBCO Object Service Broker text-based processing, conversational transactions use the DISPLAY statement.

CR

See *change request*.

current group

The security group from which a user is currently operating. A user's current group is specified in the user profile. When a user attempts to access an object, if the user's user ID is not listed in the permissions for that object, the name of the user's current group is the next item looked for in the permissions for the object.

D

database

The store of data associated with a specific node, also known as the Pagestore. Each database has its own Data Object Broker. See also *Pagestore*.

database server

See *external database server*.

Data Object Broker

The Data Object Broker manages updates to the MetaStor and Pagestore, performs Pagestore accesses and updates on behalf of transactions running in Execution Environments, controls requests to external data servers, and controls data integrity. In a distributed environment, it is identified by its node name. The Data Object Broker was formerly referred to as the Control Region.

data table

A set of occurrences (rows), each uniquely identified by a primary key.

data type

The meaning of the data stored in the field. The data type, also referred to as the semantic data type, determines the conversions and allowed operations that can be done on the data, and implies some TIBCO Object Service Broker edits on values placed in these fields.

default permissions list

As used by Security, a default permissions list names members (users or security groups) and their default access rights to new objects created by a user or group. You can specify a default permissions list for a user ID, for a Security group, or for all users.

Dependent Region

See *Execution Environment*.

derived field

A field whose value is not explicitly stored in the database. The value can be determined through a reference to another field, a rule call, or a report function. For example, the value in a field can be derived from the totalling of values in another field.

digit placeholders

A set of characters that holds display space for numeric fields. Each character has a specific representation, and represents one digit of a numeric field.

display mask

A set of display conventions applied to numeric or date fields. For example, applying the display mask MM/DD/YY to the date 2000-01-01 causes it to appear as 01/01/00.

disposition

As used in Promotions, this refers to whether a TIBCO Object Service Broker object is designated for creation, replacement, or deletion in a target system.

distributed data

The placement and access of data on separate TIBCO Object Service Broker systems (nodes). See *distributed environment*.

distributed environment

An application environment where data used in an application can be stored on a TIBCO Object Service Broker system (node) separate from the node where it is being accessed. Communications must be previously established between the nodes.

DOB

See *Data Object Broker*.

DOB name

The name of a Data Object Broker. It identifies a TIBCO Object Service Broker system and is equivalent to the TIBCO Object Service Broker node name. See *node name*.

domain

1) The TCP/IP network name that your network administrator defined and configured. 2) In SNA, the system services control point providing system services for its owned resources such as lines, terminals, and applications.

E

E

Used for shareable tools to denote that a tool is executable. See also *C* and *F*.

EE

See *Execution Environment*.

enable

As used by Security, an action taken to activate previously defined permissions for an object set.

entry rule

1) The first rule executed at the start of a text-based application.2) As used by Promotions, the first rule that is to be executed for a set of rules. This set of rules is executed and then discarded as part of a promotion.

event rule

A rule that validates access or triggers additional processing when the data in a table is accessed. It is associated with the table definition. See *trigger rule* and *validation rule*.

exception

A condition raised as a result of circumstances occurring during rules processing. An exception causes the flow of a program to change and can be handled by a program by the inclusion of an ON *exception* statement. Exceptions can be system-defined or user-defined.

exception handler

A defined sequence of actions to be executed in a rule if an exception is encountered during rules processing.

Execution Environment

The TIBCO Object Service Broker transaction manager. It handles application programming tasks such as rules execution, screen and report I/O, and I/O to external data. It is dependent on the Data Object Broker for access to the MetaStor. Formerly referred to as the Dependent Region.

export table

A table that is used to export data from TIBCO Object Service Broker to a sequential file or data set. It has a table type of EXP.

expression

A combination of elements and arithmetic or concatenation operators that results in a single value when evaluated.

expression element

A local variable, field of a table, rules argument name, function call, constant, or indirect reference.

external routine

See *routine*.

external database gateway

A self-contained program provided with TIBCO Object Service Broker that allows concurrent real-time access to external data from within TIBCO Object Service Broker. It is associated with and managed by a specified Data Object Broker.

external table

A generic term to describe a table based on information from an external source.

extracted object

As used by Promotions, an object that is copied from a source system and stored in a file pending promotion to a target system. Objects are extracted to prepare them for their promotion to multiple TIBCO Object Service Broker systems. After extraction, the objects are still in the source system.

F

F

Used for shareable tools to denote that a tool is a function. See also *C* and *E*.

Fail Safe processing

A commit process used to ensure that updates made in a single transaction to multiple databases (owned by one or more TIBCO Object Service Broker systems, and optionally one external database server) are synchronized at commit time.

field definition

A collection of characteristics for a field, such as its name, semantic data type, and syntax.

field definition area

The section of the Screen Table Painter or Report Table Painter screen used to define data fields.

fill character

The display character used to pad a screen field for the purpose of the display, for example, a dot (.).

fill string

The character string used to fill in a portion of a report field currently containing null values.

filter

A predefined view of security audit data. Filters are used from within the Audit Log facility.

fixed columns

The left-most <*nn*> columns of a scrollable screen table that do not scroll horizontally.

folding

The mapping of lowercase characters to uppercase characters, such as for purposes of case-insensitive string comparison.

footer page

The end page of a report. It is normally created using a break table.

function

A rule or routine that returns a value.

G

gateway

See *external database gateway.*

global field

A field defined by a system administrator to be used organization-wide when tables are defined. Global field definitions are accessed and incorporated into applications from within the various definer tools.

group rights

As used in Promotions, a set of rights that permits all the members of a group operating out of the same library to share the management of rights and change requests among themselves.

I

ICU

International Components for Unicode–a mature, widely used set of C/C++ and Java libraries for Unicode support. Refer to the *ICU web site.*

IDgen specification

An attribute specified in the Table Definer that determines whether the user provides a unique primary key value for each occurrence of a table (IDgen=N), or TIBCO Object Service Broker generates a value for the primary key field of each occurrence (IDgen=Y).

image area

The section of the Screen Table Painter or Report Table Painter used to paint literal text and data fields. It visually represents a screen table or a report table.

import table

A table that is used to import data from an external environment into TIBCO Object Service Broker. It has a table type of IMP.

indirect reference

A method used when writing rules that enables the identification of objects (tables, screens, rules, and fields) to be deferred until runtime. This allows table names and field references to be stored in tables and/or to be obtained from rules arguments.

in-doubt transaction

A transaction involving an external DBMS or peer TIBCO Object Service Broker system that could not be completed. In-doubt transactions are kept in the contingency log until the update is confirmed by the other system(s), then written to the redolog.

installation library

A rules library that contains production rules developed by your installation. SITE is the default name of this library.

instance

1) A member of a class of objects. For example, the EMPLOYEE table is an instance of a table object.
2) In a table with data parameters, all the data associated with a particular set of parameter values. For example, #RECEIPTS(79912, 06-03-1998) is an instance of the #RECEIPTS(employee#, batchdate) table.

instantiate

To create an explicit instance, that is, to assign values to its variables.

J

SDK (Java)

A client facility that allows programs written in Java to access TIBCO Object Service Broker. The SDK (Java) client can be in a different address space than the server, which is the SDK (C/C++) server. See also *Call Level Interface* and *SDK (C/C++)*.

L

lexical elements

Identifiers (including reserved words), numeric literals, string literals, and delimiters.

library

1) As used in Promotions, the grouping method used to implement group promotion rights. 2) See *rule library*.

line command

A command used in the line command field denoted by an underscore (_) at the left or right of each line. These commands perform various functions on an object in the list, depending on which screen you are using.

literal field

A field that cannot be modified by a user, contains only text, and cannot have data inserted into it.

local library

A rules library that contains rules written by application developers.

local variable

A variable declared within a rule. Its scope is the rule where it is declared and descendant rules in the same transaction.

locale

The portion of a user's environment that is sensitive to the cultural conventions of a particular language, country, or territory. The value defined for the locale establishes the language used for data manipulation and storage within a Data Object Broker.

M

MAP table

A table used to map the contents of memory to fields. Before a MAP table is instantiated, its size and scope must be pre-allocated by a TIBCO Object Service Broker application request.

mask

See *display mask*.

member

As used by Security, a user or group that is identified by a unique name.

menu

A list of options created through DEFINE_MENU that are presented on a screen. A menu can be part of an application, a login screen for users, or an individualized login screen.

menu definition

A collection of attributes for a menu, including its name, fields, headings, and option titles.

message line

A line at the bottom of TIBCO Object Service Broker screens. Messages generated by the system, and by user-written rules, can appear on this line.

message log

A collection of messages generated by a transaction. There are two types of message logs: a system log and a user log containing output directed by a user-written rule.

MetaStor

An active store for all metadata: all definitions, characteristics, access paths, and storage locations of all objects in TIBCO Object Service Broker. Each Data Object Broker has its own MetaStor.

N

Native Execution Environment

In z/OS environments, this refers to a VTAM- and TCP/IP-based Execution Environment. In non-z/OS environments, this is another name for the Execution Environment that provides online transaction processing. See *Execution Environment*.

node name

The name associated with a Data Object Broker. In a distributed environment it is used to identify a Data Object Broker to another Data Object Broker.

non-display field

A field defined to a report or a screen in such a way that it does not display on the report or screen. It can be used to store data used by other fields and rules.

non-restrictive environment

A development environment where users can save modifications they made to an object, even if the promotion rights of the object are held by someone else.

O

object

An instance of an entity recognized by TIBCO Object Service Broker, such as libraries, screens, reports, tables, and rules.

Object Browser

A tool that allows you to browse a predetermined list of objects.

Object Manager

A tool that displays the contents of a table (usually a list of objects), and allows a customized set of commands to operate on the display.

object set

A collection of TIBCO Object Service Broker objects grouped together using the Object Set Definer, which can be managed as a single entity. It can be treated as a single object for the purposes of defining security and in selection criteria for promotions. An object set can contain a TIBCO Object Service Broker object, including other object sets.

occurrence

A row of a table.

OCS

See *TIBCO Object Service Broker Communications Support*.

OIG

An acronym for Object Integration Gateway.

operator

1) An entity that can be applied to one or more operands to yield a result. 2) The person responsible for monitoring system operations and communicating with the operating system.

operator console

A display console that is used by an operator to communicate with the system. It is used to monitor system operations and specify information about application programs and I/O operations.

options area

The section of the Report Generator screen that displays valid options for each field on the screen in succession. You can select any of these options as values for a field. This section appears in the lower portion of most Report Generator screens.

orphan page

A page in the Pagestore that is not available for use and that is not referenced by a table.

ostty

The graphical presentation of the TIBCO Object Service Broker workbench.

owner

As used in Security, each object in TIBCO Object Service Broker has a user ID assigned as its owner, initially the creator of the object. The owner can access and delete the object, manage permissions to the object, and transfer ownership. System administrators and the

owner's security administrator share ownership privileges with the owner.

P

page
A fixed length 4096 byte area that the Data Object Broker uses for data storage.

page data set/page file
A data set (on z/OS) or a file (on Windows or Solaris) that holds TIBCO Object Service Broker data. Page data sets and page files are grouped within individual segments and stored in the Pagestore.

Pagestore
The data store for all the segments associated with a specific Data Object Broker.

parameter

A value that logically partitions data in a table. In a distributed data environment, it is also used to partition data by location (TIBCO Object Service Broker node).

parameter value table
A type of TIBCO Object Service Broker table that lists the different data parameter values for a given table. The table type is PRM.

peer TIBCO Object Service Broker
A Data Object Broker that communicates with one or more other Data Object Brokers in a distributed data environment.

peer server
A server used in a distributed data environment to connect peer TIBCO Object Service Broker systems.

permissions
As used in Security, the object type specifies access rights to an object. These access rights are specified by designating which subset of access modes are to be allowed on the object or type.

primary command
A command used in the primary command field of a screen. These commands perform various functions on the entire screen, depending on which screen you are using.

primary command field
The field where primary commands are typed. See also *line command*.

primary key
A field whose values uniquely identify occurrences of a table. See also *composite primary key*.

print field
A field whose values you want to print on a report. Normally used in conjunction with a break field.

procedure
A rule or routine that does not return a value.

promotion administrator
A person responsible for administering the promotion of application code, and applying the changes submitted by developers.

pseudo-conversational transaction
A transaction started using the DISPLAY & TRANSFERCALL statement that allows the data displayed for a screen and the screen context to be maintained, but terminates the current transaction and starts up a new transaction. To the user, it appears to be a conversational transaction. Pseudo-conversational transactions are used to minimize resource usage.

R

reference field

A field that references another table to verify values that a user can insert, or in the case of user or automatic prompting, displays values from which a user can select.

reference table

A table whose primary key is used as verification for values that a user can insert or add to another table where the reference table is specified in the definition.

refresh disposition

As used in Promotions, in the case of a disposition mismatch this allows the change request being promoted to take on the disposition of the target system.

SDK (C/C++)

A set of client and server facilities that enable programs written in third-generation languages to access TIBCO Object Service Broker. The server portion of the interface can be in a different address space than the client portion. See also *Call Level Interface* and *SDK (Java)*.

repackaged

A term sometimes used in Promotions to describe a consolidated change request.

report definition

A collection of attributes for a report that includes its name, component report tables, and the attributes of the report tables.

report field

A field in a report table.

report table

A table used to present report data. It is defined to a report and can be shared with other reports. The report table can be a title report table, a body report table, or a break table. It has a table type of RPT.

resident table index

A stored list of all the tables defined in TIBCO Object Service Broker that contains pointers to where more information is stored about the data and the indexes for the tables. It is abbreviated as RTIX.

resource management

The management of the TIBCO Object Service Broker network configuration.

restrictive environment

As used in Promotions, a development environment where only one person (user-restrictive), or one group (group-restrictive), has promotion rights to an object. In a user-restrictive environment, objects held by the user cannot be modified by other users. In a group-restrictive environment, objects held by the group cannot be modified by others outside of the group.

rights

As used in Promotions, the permission provided by TIBCO Object Service Broker to a user or group to modify an object after the object is promoted. Only user IDs with borrower status can obtain rights to an object. The type of development environment determines the rights enforced. See also *restrictive environment*.

rollback

During transaction processing, an action that causes changes to the database to be discarded. The database is returned to the state it was in at the previous committed action. This action occurs as part of abnormal termination of a transaction, or can be explicitly initiated with the ROLLBACK statement.

routine

A program that is written in a programming language other than TIBCO Object Service Broker rules. It can be called from within a rule. It is also known as a builtin.

RTIX

See *resident table index*.

rule

A unit of code written in the rules language that can be invoked by itself or by another rule.

rules-based gateway

An external database gateway written in the rules language, for example, the TIBCO Service Gateway for CA-Datacom. It operates from within a Native Execution Environment.

rule definition

The name of the rule, any arguments, and the declaration of local variables.

rules language

The TIBCO Object Service Broker programming language. A high-level language that consists of a set of data access, assignment, and control statements.

rules library

A TIBCO Object Service Broker library that contains rules. The three types of rules libraries are: local, installation, and system. The user session options determine the search order for the libraries.

S

screen

A TIBCO Object Service Broker-defined screen built from screen tables. It appears in text mode.

screen definition

A collection of attributes for a screen that includes its name and component screen tables.

screen field

A field in a screen table.

screen table

A table used to display data on a screen. It is defined to a screen and can be shared with other screens. It has a table type of SCR.

scroll field

The field in a screen table used to contain a scroll amount.

SecAdmin

See *security administrator*.

secondary index

A structure for retrieving table occurrences based on the values in a secondary key field rather than the primary key. This can improve performance because only the secondary index structure is searched instead of the entire table.

secondary key

A predefined field (or set of fields) that can be used to retrieve data through a secondary index. A secondary key is not used to order data.

security administrator

As used in Security, a privileged user who shares ownership of objects with his or her subjects. Security administrators are usually allowed to create, update, and delete user IDs, depending upon the authorities specified in their SecAdmin profile. See *subject*.

security group

As used in Security, a set of users who, during a TIBCO Object Service Broker session, can

operate out of any group in which they have membership. They can also choose not to operate out of a group. When users operate out of a security group, they are able to share objects and share access rights to those objects unless access is specifically overridden by the security permissions for their user ID.

segment

An explicit grouping of page data sets/files within the Pagestore. A segment holds only TDS data. Segment 0 holds the MetaStor.

semantic data type

See *data type*.

server

See *external database server*.

service provider

A peer Data Object Broker or an external database server.

session

A unit of resources allocated to a user when the user connects to an Execution Environment and Data Object Broker, thereby enabling the user to run transactions in TIBCO Object Service Broker. The resources are released when the user exits from TIBCO Object Service Broker.

session manager

A TIBCO Object Service Broker program that enables a user to run one or more transactions. The transactions can be chosen from a TIBCO Object Service Broker developer workbench, administrator workbench, or a user-defined menu.

session menu

See *workbench*.

session options

User specifiable options that determine the characteristics of a TIBCO Object Service Broker session. In most cases, default values apply if no user specification is made.

session table

A memory-resident table whose data persists for the life of a TIBCO Object Service Broker session. It has a table type of SES.

shareable tools

A class library of actions or methods for use in application development. These are developed by and shipped with TIBCO Object Service Broker.

SIX

See *secondary index*.

sort fields

The report fields used to determine how data is to be sorted.

source system

As used by Promotions, the system from which promoted objects are being promoted.

statement number

A number displayed in the system message log that identifies the rules statement where an error occurred. Statement numbers disregard action sequence numbers and refer to each statement in the rule.

subject

As used by Security, a user under the responsibility of a given security administrator. Security administrators have ownership privileges to all objects owned by their subjects.

subview table

A table that is based on the data of another table (a source table). It can consist of selected fields of the source table, derived fields, or both. It has a table type of SUB.

summary field

The field used to determine how data is to be summarized. A new report line is printed for every change in value of a summary field.

summary report

A type of report where a print line is produced when a summary field changes its value. In a non-summary report, a print line is produced for every body table occurrence.

surrogate pair

A Unicode representation of a single abstract character that consists of a sequence of two 16-bit code units.

syntactic type

See *syntax*.

syntax

The format used to store the data for a field.

system administrator

1) As used by Security, a user with the highest security clearance (level 7). The system administrator has full access to all objects in the system, can grant or revoke access privileges, add, delete, or change users, and appoint security administrators and other system administrators. 2) An individual who maintains the TIBCO Object Service Broker system.

system library

A rules library that contains system rules developed by and shipped with TIBCO Object Service Broker. COMMON is the default name of this library.

T

table

See *data table*.

table buffer

Interface through which a rule can access and update an occurrence in a table. Information is placed in the buffer with assignment, GET, or FORALL statements.

table definition

The description of a table including its type, parameters, if any, field names and attributes, and event rules, if any.

table instance

All table occurrences that are associated with a specific parameter value (for a table with a single parameter), or a specific set of parameter values (for a table with multiple parameters).

table.field notation

The notation used within rules to identify a specific field within a specific table when processing data. This is the reference notation used for all the types of TIBCO Object Service Broker fields and tables.

target system

The system to which objects are being promoted.

TDS (Table Data Store) table

A type of table used to store data in the TIBCO Object Service Broker database. The MetaStor is stored as TDS data. TDS is the default table type.

Telnet 3270

A protocol that is a subset of the Telnet protocol, which provides a general bi-directional, eight-bit byte oriented means of communication. It is primarily used to connect terminal devices with terminal-oriented processes. The name Telnet

3270 protocol refers to the implementation of transferring 3270 display information using Telnet capabilities.

temporary table

A table whose data is available for the life of a transaction. At the end of the transaction, the data is discarded. It has a table type of TEM.

test mode

A state that causes changes to tables that are initiated by a user-written rule to be in effect only for the duration of the transaction. When the transaction ends or a COMMIT is issued, the updates are discarded.

TIBCO Object Service Broker Communications Support

A component of TIBCO Object Service Broker that presents a single, consistent, and general purpose Application Program Interface (API) to higher-level TIBCO Object Service Broker components that require the use of varied communications facilities. Formerly called Huron Communications Support (HCS).

TIBCO Object Service Broker UI

An Eclipse-based graphical interface for creating and modifying TIBCO Object Service Broker and Object Integration Gateway objects.

title literals

Literal text that is contained in a title row of a screen table or a report table.

title report table

The table that comprises the top or bottom titles of a report. Title report tables bracket the body report table but are defined independently of it.

title rows

The first <*nn*> rows of a screen table that do not scroll vertically.

token

An item in a group of lexical elements.

tool

A procedure or function that extends the functionality of the rules language.

transaction

A unit of processing that acquires locks, allows database synchronization points within the processing unit with the COMMIT and ROLLBACK statements, and binds object definitions until the end of the processing unit. Resources are released when the transaction ends.

transaction level

The depth at which a transaction is nested. The first nested transaction is at level two. Transactions executed by another transaction are nested one level deeper, that is, higher in number, than their parent.

transaction stream

See *transaction level*.

trigger rule

A rule that allows additional processing to take place whenever a table is accessed.

U

Unicode

Refer to the Unicode web site at *www.unicode.org*.

Unicode Transformation Format (UTF)

An encoding scheme for Unicode. TIBCO Object Service Broker uses UTF-16BE (Big Endian) internally. In this encoding, characters occupy either 2 or 4 bytes. All characters in common usage occupy 2 bytes in this scheme. Other

transformations are supported as external syntaxes.

update mode

A method of accessing tables so that data is locked to maintain data integrity when changes are made to the data.

user

A person or entity, such as another process, accessing TIBCO Object Service Broker through the Execution Environment or Display Client.

user-defined event

An event defined by the application developer to be triggered by a user action. A user-defined event can call a shareable tool or be redirected to a system-defined event. These events are used in graphical applications.

V

validation rule

A rule that verifies whether a specific table access or screen input is allowed.

W

workbench

A standard menu that groups a number of TIBCO Object Service Broker tools and user applications, in any combination, so that a user has ready access to them. There are two standard workbenches supplied with TIBCO Object Service Broker: STANDARD and @ADMIN. Custom workbenches can be created using the DEFINE_MENU tool. A user's security profile determines which workbench appears when the user logs in to TIBCO Object Service Broker.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X