TIBCO® Object Service Broker for Open Systems

Utilities

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 vii

Related Documentation viii
TIBCO Object Service Broker Documentation viii
Typographical Conventions xiii
Connecting with TIBCO Resources xvi
How to Join TIBCOmmunity xvi
How to Access All TIBCO Documentation xvi
How to Contact TIBCO Support xvi

Chapter 1 Using TIBCO Object Service Broker Utilities 1

Introduction 3
What are TIBCO Object Service Broker Utilities? 3
The TIBCO Object Service Broker Database Administrator Tool 3
hrnbrbal (Segment Balance) 4
hrnbrclr (Batch Table Clear) 9
hrnbrext (Extract Selected Pages) 10
hrnbrial (Move ACCESSLOG) 11
hrnbrnls (Translate File Between Different Code Pages) 14
hrnbrpgc (Pagestore Correction) 18
hrnbrptr (Batch Pointer Check) 20
hrnbrset (Batch Segment Re-initialization) 26
hrnbrsix (Batch Secondary Index Build – TDS Tables) 30
hrnbrtbl (Batch Load) 33
hrnbrula (Batch Unload from Archive) 40
hrnbrulb (Batch Unload (Offline)) 48
hrnbrulh (Batch Unload (Online)) 54
hrncr (TIBCO Object Service Broker Data Object Broker) 61
hrncrSvc (Install Data Object Broker as a Windows Service) 66
hrnspjex (Journal Extraction (or Journal Spin)) 68
hrnspset (Reset Journal) 70
hrntladm (Administration Menu) 72
hrntlbps (Backup Pagestore) 74

hrntlfcl (Format Contingency Log) . . . 76
hrntlfjr (Format Journal) . . . 79
hrntlfps (Format Pagestore) . . . 82
hrntlfrl (Format Redolog) . . . 85
hrntlmrg (Journal Merge) . . . 87
hrntlpcl (Print Contingency Log) . . . 89
hrntlrps (Restore Pagestore) . . . 90
htrans (Translates z/OS Files To and From TIBCO Object Service Broker Binary Files) . . . 93
osBatch (Start a TIBCO Object Service Broker Batch Client) . . . 94
osMonSvc (Install osMon as a Windows Service) . . . 98
rsview (Data Object Broker Resource Viewer) . . . 99

Appendix A TIBCO Object Service Broker Files . . . 101

Description of the Files . . . 102
Data Object Broker Parameter Files . . . 102
Database Definition . . . 102
Data Object Broker Log File . . . 105
TIBCO Object Service Broker Directory . . . 106

Appendix B Defining Batch Load Control Cards Manually . . . 109

Overview . . . 110
Manually Defining Control Cards . . . 110
Keyboard Constraint . . . 110
Specification Cards . . . 111
Field Values . . . 111
Page Fill Tailoring . . . 113
Tailoring Guidelines . . . 113
Page Split/Merge Processing . . . 113
Input Definition Cards . . . 114
Field Values . . . 114
Output Definition Cards . . . 116
Field Values . . . 116
Value Cards . . . 118
Field Values . . . 118

Appendix C Null Handling . . . 119

Syntax Specific Nulls . . . 120
Alternative Null Equivalents . . . 122
-O Option (Batch Unloads hrnbrulb and hrnbrulh Only) . . . 123

Floating Point Nulls 123

Index 125

Preface

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

This manual contains an alphabetically ordered listing of TIBCO Object Service Broker utilities for Open Systems systems. These are TIBCO Object Service Broker administrator utilities that are typically executed from a command line.

Topics

- Related Documentation, page viii
- Typographical Conventions, page xiii
- Connecting with TIBCO Resources, page xvi

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 Using TIBCO Object Service Broker Utilities

This chapter lists and describes the utilities you can use to manage the TIBCO Object Service Broker database and other files.

Topics

- Introduction, page 3
- hrnbrbal (Segment Balance), page 4
- hrnbrclr (Batch Table Clear), page 9
- hrnbrial (Move ACCESSLOG), page 11
- hrnbrnls (Translate File Between Different Code Pages), page 14
- hrnbrpgc (Pagestore Correction), page 18
- hrnbrptr (Batch Pointer Check), page 20
- hrnbrset (Batch Segment Re-initialization), page 26
- hrnbrsix (Batch Secondary Index Build – TDS Tables), page 30
- hrnbrtbl (Batch Load), page 33
- hrnbrula (Batch Unload from Archive), page 40
- hrnbrulb (Batch Unload (Offline)), page 48
- hrnbrulh (Batch Unload (Online)), page 54
- hrncr (TIBCO Object Service Broker Data Object Broker), page 61
- hrncrSvc (Install Data Object Broker as a Windows Service), page 66
- hrnspjex (Journal Extraction (or Journal Spin)), page 68
- hrnspset (Reset Journal), page 70
- hrntladm (Administration Menu), page 72
- hrntlbps (Backup Pagestore), page 74
- hrntlfcl (Format Contingency Log), page 76

- hrntlfjr (Format Journal), page 79
- hrntlfps (Format Pagestore), page 82
- hrntlfrl (Format Redolog), page 85
- hrntlmrg (Journal Merge), page 87
- hrntlpcl (Print Contingency Log), page 89
- hrntlrps (Restore Pagestore), page 90
- htrans (Translates z/OS Files To and From TIBCO Object Service Broker Binary Files), page 93
- osBatch (Start a TIBCO Object Service Broker Batch Client), page 94
- osMonSvc (Install osMon as a Windows Service), page 98
- rsview (Data Object Broker Resource Viewer), page 99

Introduction

What are TIBCO Object Service Broker Utilities?

You use the utilities to manage the TIBCO Object Service Broker database and other files.

The TIBCO Object Service Broker Database Administrator Tool

Using the TIBCO Object Service Broker Database Administrator tool, you can perform some of the same Data Object Broker configuration tasks, such as:

- Database backup and restore
- Formatting and adding new segments
- Resizing existing segments

Refer to *TIBCO Object Service Broker for Open Systems Installing and Operating* for details about starting the TIBCO Object Service Broker Database Administrator tool; and to the TIBCO Object Service Broker Database Administrator online help.

hrnbrbal (Segment Balance)

Syntax hrnbrbal [-A *auditreport*] [-p *#pages*] [-s *segment#*][-v]-f *#files* -N *new_archive old_archive*

Platforms Windows, Solaris

Description The hrnbrbal (Segment Balance) utility reads a TIBCO Object Service Broker segment archive and creates a new archive that matches the new specifications provided for the segment. This utility works at the page level only. The utility accepts archives from any of the supported TIBCO Object Service Broker platforms and is release independent.

Purpose hrnbrbal automates the redistribution of the used pages of a specified segment into a redefined space. This redistribution makes changing space allocation for a segment easier and in some cases also leads to improved performance.

We strongly recommend that the procedures described here be performed by an experienced TIBCO Object Service Broker system administrator who is familiar with the site backup and recovery environment. All the steps in these procedures should be read and understood before proceeding with this activity. All aspects of this topic should be considered, such as:

- Do you use the continuous backup method to create backup files?
- Do you use the utility hrntlbps to create backup files?
- Do you use a non-TIBCO Object Service Broker utility to create backup files?
- Do you create one backup file for all segments?
- Do you create one backup file for each segment?

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this option is omitted, the report is written to *stdout*.
-p *#pages*	Number of defined pages in the old segment definition. Default: 400000
-s *segment#*	The number of the offline segment. Default: 0 (this is the base segment)

Argument	Description
-v	Produce a verbose audit trail report that gives complete details of all modifications made to the segment, for example, the old and new numbers of all pages. Default: verbose not required
-f *#files*	The number of files in the newly defined segment.
-N *new_archive*	The path to and name of the new archive (output).
old_archive	The path to and name of the old archive (input).

Tasks to Complete Before Executing hrnbrbal

Task A Take a Journal Spin

1. Vary offline the segments on which you are to run hrnbrbal.

   This segments is referred to as the target segment.

2. Force a journal spin.

   OR, if *all* segments are to be balanced:

3. Suspend all users.
4. Force a journal spin.
5. Shut down the Data Object Broker.

Task B Take a Backup

1. If you are using continuous backup, run a spin merge and create a new continuous backup file for the target segments.
   OR
   If you do not use continuous backup, run hrntlbps (Backup Pagestore) to take a backup of each target segments.
2. Run hrnbrptr (Batch Pointer Check) on each target segment.
3. Check that the resulting audit report contains no errors or orphan pages. Fix the errors and recover all orphan pages before proceeding.

Steps to Run hrnbrbal

1. Run hrnbrbal (Segment Balance) for each target segment.

   Ensure you are using the appropriate archive file for each segment.

2. Run hrnbrptr (Batch Pointer Check) against each newly balanced segment archive to verify its contents.

Ensure you are using the appropriate newly balanced archive for each segment.

3. Check that the resulting audit report contains no errors or orphan pages.

If errors occur at this point, call TIBCO Support immediately.

Steps to Take After Successfully Executing hrnbrbal

When the new archive is created:

1. If the number of files in a target segment changed, modify the ACBS statement for that segment in your dbdef file to equal the new number of files in the segment.
2. Delete the existing files in the target segments.
3. Run hrntlfps (Format Pagestore) to format each new segment definition.
4. Run hrntlrps (Restore Pagestore) to restore the newly balanced archive to the segment, using the output from hrnbrbal as input to hrntlrps.
5. If you are using continuous backup, restart it from this point for each target segment using the output of hrnbrbal to create a base backup for each segment. Alternatively, you can run hrntlbps (Backup Pagestore) to create the base backup.
   OR
   If you are not using continuous backup, run hrntlbps (Backup Pagestore) to take a full backup of each target segment.
6. For each target segment, delete all journal data that exists in the spin, merge, and if you use continuous backup, the backup files.

Failure to complete this step can result in a corrupted target segments and loss of data.

7. Restart or recycle the Data Object Broker.

See Also

TIBCO Object Service Broker for Open Systems Managing Backup and Recovery for additional information about the backup and restore procedures.

Return Code Settings

Return Code	Meaning
0	No errors.
4	Warnings issued.
200	No audit file available.

Return Code	Meaning
201	Non-recoverable error.

Example The example command that follows reads the segment archive s0f1.bak and creates the new archive s0f3.bak, whose pages are balanced over three files:

```
hrnbrbal -A s0f3bal.aud -f 3 -N s0f3.bk s0f1.bak
```

This example creates the following audit report (s0f3bal.aud):

```
hrnbrbal   Balance Segment number:       001   2007 April 17   15:25     v1.0

Balance Archive s0f1.bak into Redefined Archive s0f3.bak
Number of files in redefined segment: 003

PHASE 1 - Analyze Old Archive:

Segment: 0000; Old File: 0; Extent: 0; Page# 00000000
# pages available: 19767; Bit map size: 09A7

File 0 High Used Page (not including BITMAP pages): 00004D36; # Used Pages: 19766; #
Pages Ignored: 1

OLD ARCHIVE - Read: 19767; Processed: 19767; Used: 19766; Ignored: 1


PHASE 2 - Re-assign Page Numbers:
Pages to be assigned to each of 003 files: 6589
Additional pages to be assigned to the 1st file: 2

New File #: 0; Extent: 0
High Used Page (not including BITMAP pages): 000019BE; # Assigned Pages: 6590; #
Control Pages  1


New File #: 1; Extent: 0
High Used Page (not including BITMAP pages): 010019BC; # Assigned Pages: 6588; #
Control Pages  1

New File 2 Summary;
High Used Page (not including BITMAP pages): 020019BC; # Assigned Pages: 6588; #
Control Pages  1



PHASE 3 - Create New Archive:

New Control Page: 00000000
Prev: FFFFFFFF; Next: FFFFFFFF; Type: x'F0'
Number of Pages: 6591; Bit Map Size: 0338; File High Page: 000019BE

New Control Page: 01000000
Prev: FFFFFFFF; Next: FFFFFFFF; Type: x'F0'
```

```
Number of Pages: 6589; Bit Map Size: 0338; File High Page: 010019BC

New Control Page: 02000000
Prev: FFFFFFFF; Next: FFFFFFFF; Type: x'F0'
Number of Pages: 6589; Bit Map Size: 0338; File High Page: 020019BC

Bytes: Input 15335528; Output - Old 15331464; New 12192


OLD Pages - Read: 19767; Processed: 19767; Used: 19766; Ignored: 1
```

hrnbrclr (Batch Table Clear)

Syntax hrnbrclr [-s *segment#*] [-A *auditreport*] *tablename*

Platforms Windows, Solaris

Description The hrnbrclr (batch table clear) utility deletes all data from a TDS table. This offline utility operates in the same way as the online $CLRTAB tool.

Online transactions are subject to the chosen transaction limits of your site. If you exceed your online transaction limit, you can use the hrnbrclr utility to clear the table while it is offline.

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this option is omitted, the report is written to the screen.
-s *segment#*	The offline segment containing the specified table. If this option is omitted, the segment number defaults to zero (that is, the base segment is used).
tablename	The name of the table to be cleared.

If you are clearing a table located in the base segment, the Data Object Broker must first be shut down.

Constraints The following constraints apply:

- The segment containing the specified table must be offline.
- The segment must be formatted for TDS data.
- The hrnbrclr utility cannot delete data from a table instance of a parameterized table. To delete data from a selected table instance, use $CLRTAB.

hrnbrext (Extract Selected Pages)

Syntax

hrnbrext [*arguments*] *backupfile* [< ***inputfile.txt***]

Platforms

Windows, Solaris

Description

`hrnbrext` is used to extract selected pages from a pagestore to assist with operational maintenance. It should be used only under instructionsfrom TIBCO Support.

The page numbers to be extracted can be entered manually once the utility has been invoked, as 1 to 8 character hexadecimal numbers each on a separate line, with input being terminated by <CTRL-Z> on Windoes, or <CTRL-D> on Solaris.

Alternatuively, they can be entered on separate lines within a text file and piped to the utility (as per ***inputfile.txt*** in the above syntax)..

Arguments

Argument	Description
-s *segnumber*	Segment from which the page will be extracted,
-w	Overwrite ***backupfile*** it it exists..

Environmental Variables

OS_ROOT

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (`%OS_ROOT%\database\huron.dir` in Windows, or `$(OS_ROOT)/database/huron.dir` in Solaris).

hrnbrial (Move ACCESSLOG)

Syntax hrnbrial [-A *auditfile*] [-s] *segmentnumber*

Platforms Windows, Solaris

Description The hrnbrial utility moves the ACCESSLOG table (Audit log), which stores the security audit log, from the MetaStor (segment 0) on which it is shipped, to another segment. The hrnbrial utility can be used as part of the process for installing TIBCO Object Service Broker, or prior to promoting a Pagestore into production.

See Also *TIBCO Object Service Broker for Open Systems Installing and Operating* for more information on when to invoke the hrnbrial utility.

Prerequisites Before invoking the hrnbrial utility, ensure that the following prerequisites are satisfied:

- Both segment 0 and the target segment are backed up. For information on backing up a segment, refer to *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*
- Since this utility directly modifies both segment 0 and a target segment, whoever submits it for execution must have write authority to both segments.
- A valid ACCESSLOG occurrence exists on segment 0, including an entry in TABLES where TABLES.SEG is not NULL and a resident table index (RTIX) entry where primary path = a segment 0 page number.
- There must be no more than one D page of data in the ACCESSLOG table. This data is not retained in the move to the new segment.
- A valid ACCESSLOG database entry exists on segment 0.
- The target ACCESSLOG segment is correctly defined in the dbdef file and has been formatted using the hrntlfps utility.
- There is no ACCESSLOG entry on the target segment.
- The ACCESSLOG table is archived or purged. For information on archiving and purging the ACCESSLOG (Audit Log), refer to *TIBCO Object Service Broker Managing Security.*
- The Data Object Broker is shut down.

Arguments

Argument	Description
-A *auditfile*	The full path and filename of the audit file. This argument has a default of stdout.
-s *segmentnumber*	The target segment number to which ACCESSLOG is being moved. This must be a valid Pagestore segment number. There is no default.

Considerations

Note the following when using the hrnbrial utility:

- This utility moves the ACCESSLOG table only from segment 0 to a segment other than segment 0. ACCESSLOG can be moved only once and after it is moved off segment 0 it cannot be moved again. If possible, move ACCESSLOG to its own segment where it can be kept separate from other data.
- Since this utility updates the segment offline and does not perform journaling, you must back up any changes you make to the target segment. For information about backing up a segment, refer to *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Examples

Windows

```
hrnbrial -A d:\audit\brial.aud -s 03
```

Solaris

```
hrnbrial -A d:/usr1/audit/brial.aud -s 03
```

These examples show:

- The audit log is written to the file brial.aud in the directory d:\audit (Windows) or d:/usr1/audit (Solaris).
- The ACCESSLOG table is being moved to segment 03.

Checking Installation

The audit file logs the activities of the utility and reports any errors or problems that occur. The audit file must be checked after running the hrnbrial utility; the message: `S6BBC915I ACCESSLOG INSTALLATION COMPLETE` indicates that the ACCESSLOG table was moved successfully.

Sample Audit Log The following illustrates a sample audit log for the hrnbrial utility:

```
*batchsrv      OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 06 TIME 12 06
Requested Utility: hrnbrial

PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - OPENED

*hrnbrial           INSTALL ACCESSLOG

PAGESTORE SEGMENT NAME METASTOR   SEGMENT # 0  TYPE TDS - OPENED

  S6BBC910I Search METASTOR for ACCESSLOG

Table's primary path begins at page: 00000294; no secondaries allocated.

  S6BBC911I Checking METASTOR pointers

  S6BBC912I Installation Commenced - Target Segment

PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1 - CLOSED

  S6BBC913I Installation Completed - Target Segment
  S6BBC914I Modifying METASTOR pointers


PAGESTORE SEGMENT NAME METASTOR   SEGMENT # 0 - CLOSED

  S6BBC915I ACCESSLOG INSTALLATION COMPLETE

                                UTILITY     SUMMARY

          TOTAL PAGES FREED         00000000002

          TOTAL PAGES UPDATED       00000000004

*          START TIME  11:06:48    END TIME    12:06:48
```

hrnbrnls (Translate File Between Different Code Pages)

Syntax hrnbrnls -A *auditfile* -C *controlfile* [*arguments*] -L *inputfile* -U *outputfile*

Platforms Windows, Solaris

Description The hrnbrnls utility is used to translate files between different code pages.

See Also *TIBCO Object Service Broker National Language Support* for a list of supported code pages.

Arguments

Argument	Description
-A *auditfile*	The file to which the audit report is written. If this argument is omitted, the report is written to stdout.
-C *controlfile*	The full path and name of the control file. This argument is mandatory.
-m	Indicates that the input file is in z/OS Variable Block format. If the input file (refer to the -L argument) was created using one of the offline batch unload utilities–hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), or hrnbrulh (Batch Unload (Online))–the -m argument must be specified on the command line. If the input file was created on z/OS and ported to Windows or Solaris, *do not* include the *-m* argument.
-w	Indicates you want to overwrite the previously existing input file. This argument is optional.
-L *inputfile*	The full path and name of the input file. This is a mandatory argument.
-U *outputfile*	The full path and name of the output file. This argument is mandatory unless -w is specified, in which case the input filename is also the output filename.
-y *inputcodepage*	The code page used to read the input file. This optional argument has a default of IBM-037.
-z *outputcodepage*	The code page used to write the output file. This optional argument has a default of IBM-037.

Constraints The following sections describe constraints associated with each of the files used in the translation process.

Control File You must prepare a correctly formatted control file (-C). This file is identical to the control file used by the hrnbrtbl (Batch Load) offline utility and can be created using either the BATCHLOAD_CARDS shareable tool, or created manually with the editor of your choice. For complete information on the use of BATCHLOAD_CARDS, refer to *TIBCO Object Service Broker Shareable Tools*.

Eligible Fields

All file fields that have all the following characteristics are eligible for translation:

- The field has an associated (target) table field defined.
- The File (input) field has a semantic type of none (blank), I, or S, and a syntax of C, V, or A.
- The Table (target) field has a semantic type of none (blank), I, or S, and a syntax of C or V.

Creating the Control File

When creating the control file, consider the following factors:

- The hrnbrnls utility is designed to translate table data that is unloaded from a TIBCO Object Service Broker table using one of the offline batch unload utilities: hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), or hrnbrulh (Batch Unload (Online)).
- The hrnbrnls utility is designed to translate unloaded table data prior to it being loaded to a TIBCO Object Service Broker table using the hrnbrtbl (Batch Load) offline utility.
- The input file (-L) for the hrnbrnls utility must be completely defined and every file (input) field must have an associated table (output) field. File fields that have no target table field definition are not eligible for translation.
- The hrnbrnls utility has no way of recognizing bit-strings. A field defined as character input that is effectively a bit-string *is* eligible for translation if its target table field is also defined as character.
- Static values are not eligible for translation.
- If a field is eligible for translation and is also a parameter or a primary key field, consider sorting the output file (-U) after translation. If any such fields exist in the input file, it is recommended that translation and sorting be done on z/OS prior to a port to Windows or Solaris if the file is to be ported in this way.

- If the file to be translated was created by one of the offline batch unload utilities—hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), or hrnbrulh (Batch Unload (Online))—it is a variable format file, in which case you must ensure that the V (variable) flag is specified on the Input File definition screen of the BATCHLOAD_CARDS tool.
- The CHARSET value in the control file has no effect on translation.

Input File (-L)

The hrnbrnls utility is designed to translate table data that is unloaded from a TIBCO Object Service Broker table using one of the batch unload utilities: hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), or hrnbrulh (Batch Unload (Online)). The unloaded file must be in TIBCO Object Service Broker data format (EBCDIC) and must be in the correct sequence (ascending: parameter 1 through *n*, and primary key 1 through *n*).

Output File (-U)

The translation creates an output file with the same characteristics as those of the input file. This file must be in the correct sequence for the table to which it is to be loaded (ascending: parameter 1 through *n*, and primary key 1 through *n*). If the sequence of the file changed with translation, and thus requires sorting prior to the load, a message indicating this is issued in the audit file.

Translate In Place (-w)

The hrnbrnls utility is designed to translate table data that is unloaded from a TIBCO Object Service Broker table using one of the offline batch unload utilities: hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), or hrnbrulh (Batch Unload (Online)). This unload file must be in the correct sequence (ascending: parameter 1 through *n*, and primary key 1 through *n*). If the sequence of the file is changed by translation, and thus requires sorting prior to the load, a message indicating this is issued on the audit file.

For Translate In Place (-w), if a problem occurs during the hrnbrnls process, it is possible that your file is no longer usable.

Audit File (-A)

This file is an activity report containing information, warning and error messages as well as run statistics. This report should always be inspected before using the output file in another process. This file is required if you encounter a problem that requires the assistance of a TIBCO Support representative.

Return Codes

Return Code	Meaning
0	Translation completed successfully.
4	At least one warning is issued.
600	Terminated abnormally.

Return Code	Meaning
602	Command line argument error.

Example

Windows

```
hrnbrnls -A d:\audit\TBL234.aud -C c:\cntl\TBL234.ctl -m
         -L TBL234.in -U TBL234.out -y IBM-037 -z IBM-278
```

Solaris

```
hrnbrnls -A /usr1/audit/TBL234.aud -C /usr1/cntl/TBL234.ctl -m
         -L TBL234.in -U TBL234.out -y IBM-037 -z IBM-278
```

These examples show:

- The audit log is written to the file TBL234.aud in the directory d:\audit (Windows) or in /usr1/audit (Solaris).
- The control file is located in c:\cntl\TBL234.ctl (Windows) or in /usr1/cntl/TBL234.ctl (Solaris).
- The input file is written in Variable Blocked format (was unloaded on Windows or Solaris).
- The input file is TBL234.in in the current directory.
- The output file is TBL234.out in the current directory.
- The data is to be translated from code page IBM-037 to IBM-278.

Related Utilities

- hrnbrula (Batch Unload from Archive)
- hrnbrulb (Batch Unload (Offline))
- hrnbrulh (Batch Unload (Online))
- hrnbrtbl (Batch Load)

hrnbrpgc (Pagestore Correction)

Syntax hrnbrpgc [-A *auditreport*] [-b] [-s *segment*] inputfile

Platforms Windows, Solaris

Description You can use the hrnbrpgc utility to reclaim orphan pages detected by the hrnbrptr (Batch Pointer Check) utility. It reads the error log from the hrnbrptr utility and designates the orphan pages as free and available for use.

We strongly recommend that you contact TIBCO Support before running this utility. Be prepared to supply output from hrnbrptr (Batch Pointer Check).

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this option is omitted, the report is written to *stdout*.
-b	Selecting the -b (browse) argument simulates an update and does not affect the Pagestore.
-s *segment*#	The offline segment number. If this argument is excluded, the segment number defaults to zero.
inputfile	Error log file created by the hrnbrptr (Batch Pointer Check) utility. This argument is mandatory.

Sample Audit Log The following illustrates a sample audit log for the hrnbrpgc utility:

```
*hrnbrpgc               PAGE CORRECTION             Date 2007 MAR 06 Time 02:18

DATE 2007 MAR 06 TIME 02:18    S6BBC107I ERRLOG WAS CREATED

S6BBP011W Request to free page: 00000066
Page detail: 00000066; Prev: FFFFFFFF; Next: FFFFFFFF; Type: T
---------> 00 970501153159 968686A285A3 0001 0014
REQUEST COMPLETE

S6BBP011W Request to free page: 01000001
Page detail: 01000001; Prev: FFFFFFFF; Next: FFFFFFFF; Type: 6
---------> 01 970501153552 01FCE20001DA 0001 002B
REQUEST COMPLETE

S6BBP011W Request to free page: 01000002
Page detail: 01000002; Prev: FFFFFFFF; Next: FFFFFFFF; Type: 6
---------> 01 970501153552 01FCE30001DA 0001 002B
REQUEST COMPLETE

S6BBP011W Request to free page: 01000032
Page detail: 01000032; Prev: FFFFFFFF; Next: FFFFFFFF; Type: G
---------> 00 970501153552 01FCE30001DA 0002 0008
REQUEST COMPLETE


                              PAGE I/O SUMMARY

          TOTAL PAGES FREED         00000000004

          TOTAL PAGES READ          00000000005

          TOTAL PAGES UPDATED       00000000000

*          START TIME  02:18:29     END TIME    02:18:29
```

Related Utility hrnbrptr (Batch Pointer Check)

hrnbrptr (Batch Pointer Check)

Syntax hrnbrptr [*arguments*] filename

Platforms Windows, Solaris

Description The hrnbrptr utility validates the integrity of page images in either an archive file or a database segment by validating the horizontal and vertical pointers within the file specified. Run this utility on a regular basis against your TIBCO Object Service Broker system backups or segments to guarantee that every referenced page is accessible.

Input The *filename* argument can be one of the following:

- The name of an archive file generated by the hrntlbps (Backup Pagestore) utility from a full system or segment backup file.

  An archive file can contain more than one segment, in which case you must run hrnbrptr against each segment individually.

- The name of a dbdef file that contains the number and location of a single live segment file. This segment must be offline and, if the segment is the MetaStor, the Data Object Broker must be shut down. When validating a live segment, hrnbrptr indicates that it is doing so on the audit report.

  The dbdef specification for the requested segment does not require the *OS_ROOT* environment variable to be set as long as the dbdef DB PATH option is used with or without FILE statements. The access path to a segment file is determined by the following: 1) the dbdef FILE statement, if specified, 2) the DB PATH option, if specified, and 3) the *OS_ROOT* environment variable.

  A segment file must have a file header at the beginning of the file. When read, this header is ignored. If it is missing, an error message is output and the utility ends.

We strongly recommend that you:

- Always run this utility following a standalone backup and after TIBCO Object Service Broker runs a continuous backup
- Regularly take backups of all segments

See Also

- *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* for information about backing up a segment
- Database Definition on page 102 for information about the dbdef file

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this argument is omitted, the report is written to *stdout*.
-d	Validate the data page header entry size. Check that the entry size is consistent with data page rows on the page.
-E *errorlogfile*	The file to which the error log is written. If this argument is omitted, the report is written to ERRLOG.ptr in the current directory.
-e #	The number of errors and warnings permitted before hrnbrptr quits. The default is 1000. A value of 0 (zero) permits an unlimited number of errors and warnings. **Note** If your archive file or segment has many problems, this setting could produce very large audit report and error log files.
-h	Print the page headers. This argument must precede the -H argument. If -H is not specified, the headers are written to PGHDR.ptr in the current directory.
-H *pghdrlist*	The file to which the page headers are written. If this argument is omitted and the -h argument is specified, the headers are written to PGHDR.ptr in the current directory. This argument has no effect unless *-h* is also specified.
-n	Optional parameter that requests the secondary index invalidated warning message be suppressed.
-O *orphanlist*	The file to which the orphan log is written. If this argument is omitted, the log is written to ORPHAN.ptr in the current directory.
-p *numpages*	The hrnbrptr utility automatically assigns a buffer of 200,000 pages for the Pagestore. Use this argument to set a higher number of pages if necessary.
-R *reflogfile*	The file to which the reference log is written. If this argument is omitted, the reference log is written to REFLOG.ptr in the current directory.
-s *segment#*	The offline segment number (not segment name). If this argument is omitted, the segment number defaults to zero (that is, the MetaStor is assumed).

Argument	Description
-W *workpath*	The path to a directory to use for temporary storage. If this argument is omitted, the current directory is used.
filename	The path of either the archive file created by the hrntlbps (Backup Pagestore) utility or a properly formatted dbdef file, which it recognizes by the .dbdef extension. For more detail, refer to Input on page 20.

Files

PGHDR.ptr

Lists the page headers of all used pages. It can be used with the REFLOG when corruption is detected. This is an optional file.

Audit report (written by default to stdout)

Provides information on each of the tables that exist in the segment, each error condition found, plus a number of summary items. It contains important information for tracing problems and should be retained. For more information, refer to Sample Audit Report on page 24.

ERRLOG.ptr

Lists error messages detected by the validation process. It provides an easy reference to help you determine if errors exist. All messages are also reported in the audit log.

This file can be used as input to the hrnbrpgc (Pagestore Correction) utility. hrnbrptr extracts the timestamp from page 1 and puts it in the ERRLOG in the form `ID=xxxxxxxxxxxxxxx`. hrnbrpgc checks this timestamp against the then current page 1 before acting on an orphan-recovery request. If there is no `ID=line` in ERRLOG or if the stamps do not match, the job aborts. If, after checking that the ERRLOG is still valid against the indicated segment, you decide to proceed with the recovery, either insert the following line after the headings in the ERRLOG file, or modify the existing `ID= line`, as appropriate: `ID=IGNORE`.

ORPHAN.ptr

Contains a list of pages where the system usage indicators did not agree with reference information detected by Pointer Check processing. Orphans *do not* appear as errors in the error count.

An orphan page is a page that is not available for use and is not referenced by a table. A page that is referenced by a table but is available for use is reported as REFERENCED BUT INDICATED FREE.

In the orphan file, the ORPHAN pages are listed with the previous and next page pointers as well as the page type. REFERENCED BUT INDICATED FREE pages are listed as a page number followed by the message text.

The orphan list file should be retained so that you can compare these results with those generated by a future run.

If your report shows ORPHAN pages or REFERENCED BUT INDICATED FREE pages, stop using the segment and contact TIBCO Support immediately. If processing continues on the segment, duplicate page errors can occur.

REFLOG.ptr

The reference log lists each referenced page in the segment. The log identifies the table, forward and backward chain pointers, and parent information. This information helps to identify how the page fits in the Pagestore.

The information in the reference log should be retained. It is intended to help the TIBCO Support identify problems if they should arise.

Sample Audit Report

The following illustrates a compressed version of an audit report for hrnbrptr:

```
hrnbrptr       POINTER VALIDATION FOR SEGMENT 001   DATE 2007 MAR 06 TIME 02:18
From Archive: seg01.bak
TABLE/SIX NAME       I/1     D/B/R   H/G     S/6     OTHERS  TOTAL   ERRORS
ARCHIVE RECORDS - Read: 0000188; Accepted: 0000188
**** SYSTEM ****     000001  000001  000000  000000  000000  000002  000000
######SEC1
  FLD02SKEY08_LAST   000000  000000  000000  000000  000000  000000  000000
  PKEY02SKEY01       000000  000000  000000  000000  000000  000000  000000
  PKEY03SKEY02       000000  000000  000000  000000  000000  000000  000000
  PKEY06SKEY03       000000  000000  000000  000000  000000  000000  000000
  PKEY07SKEY04       000000  000000  000000  000000  000000  000000  000000
  PKEY09SKEY05       000000  000000  000000  000000  000000  000000  000000
  PKEY11SKEY06       000000  000000  000000  000000  000000  000000  000000
  PKEY14SKEY07       000000  000000  000000  000000  000000  000000  000000
######SEC1          000000  000009  000000  000000  000001  000010  000000
@RULESLIBRARY        000000  000001  000000  000000  000000  000001  000000
AAAAAASEC1
  FLD02SKEY08_LAST   000000  000000  000000  000000  000000  000000  000000
  PKEY02SKEY01       000000  000000  000000  000000  000000  000000  000000
  PKEY03SKEY02       000000  000000  000000  000000  000000  000000  000000
  PKEY06SKEY03       000000  000000  000000  000000  000000  000000  000000
  PKEY07SKEY04       000000  000000  000000  000000  000000  000000  000000
  PKEY09SKEY05       000000  000000  000000  000000  000000  000000  000000
  PKEY11SKEY06       000000  000000  000000  000000  000000  000000  000000
  PKEY14SKEY07       000000  000000  000000  000000  000000  000000  000000
AAAAAASEC1           000000  000009  000000  000000  000001  000010  000000
ABIGC                000000  000001  000000  000000  000000  000001  000000
****************
GRAND TOTAL          000002  000150  000004  000000  000007  000163  000000
```

Indented entries, such as those following table AAAAAASEC1, indicate a secondary index with that secondary key field name.

Return Codes

To identify validation errors more easily, the return code is set according to the most severe error encountered:

Return Code	Meaning
0	No errors.
4	Orphan pages detected / Warnings.
8	Page header pointer error detected.
12	Duplicate page or page not in backup.

A non-recoverable error results in an abnormal termination. Refer to *TIBCO Object Service Broker Messages With Identifiers* for messages issued by this utility.

When running Batch Pointer Check as part of a batch job, have the job notify the TIBCO Object Service Broker administrator when a non-zero return code is issued. The TIBCO Object Service Broker administrator should then check the ERRLOG.ptr file to find out why the job failed.

hrnbrset (Batch Segment Re-initialization)

Syntax hrnbrset -A *auditfile* -s *segment_number*

Platforms Windows, Solaris

Description The hrnbrset (Batch Segment Re-initialization) utility re-registers empty TDS tables into any specified segment on the Pagestore (except segment 0).

This utility is useful in situations where specific data is periodically purged and reloaded into TIBCO Object Service Broker in its entirety, for example, daily sales data. The transient data is normally stored in a separate Pagestore segment. This utility can be used to:

- Rebuild the resident table index (RTIX)
- Create the empty tables
- Rebuild the secondary index structures for the deleted tables (TDS tables only)

Having used the hrnbrset utility to build the empty table and index structures, you can load/reload the data using either the LOAD tool or the hrnbrtbl (Batch Load) utility.

Prerequisite The target segment must be reformatted using the (hrntlfps (Format Pagestore) utility before invoking the hrnbrset utility.

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this argument is omitted, the report defaults to *stdout*.
-s *segment_number*	The offline segment number. This is a mandatory argument and can be any segment other than segment 0.

Constraints The following constraints apply:

- The target can be any segment except segment 0 (the MetaStor).
- The TIBCO Object Service Broker system must be active.

- The target segment (the one to be re-initialized) must be offline.

Since this utility updates the segment offline and does not perform journaling, you must back up any changes you make to the target segment. For information about backing up a segment, refer to *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery*.

Invocation

The steps to analyze, clear, and re-initialize the segment are as follows:

1. Format the segment (or re-format it if it already exists) using the hrntlfps (Format Pagestore) utility.

   If you are employing the continuous backup method, be aware that your current backup contains page images for the segment you plan to re-initialize. As a precaution, after re-initializing the segment, consider filtering out any pages associated with that segment from your latest backup. For information about filtering your backup, refer to *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery*.

2. Run the hrnbrset utility.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory under which TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location of %OS_ROOT%\database\huron.dir on Windows or ${OS_ROOT}/database/huron.dir on Solaris.

Example

Windows

```
hrnbrset -A d:\audit\auditreport -s 1
```

Solaris

```
hrnbrset -A /usr1/audit/auditreport -s 1
```

In these examples:

- The audit report is written to the file auditreport in the directory d:\audit (Windows) or /usr1/audit (Solaris).
- The target segment is segment 1.
- The hrnbrset utility connects to the Data Object Broker specified in *crparm*, which is located in the *database* directory indicated by the *OS_ROOT* environment variable.

- The hrnbrset utility builds any missing RTIX entries on segment 1. Segment 1 is located in the *database* directory, which is located in the directory indicated by the *OS_ROOT* environment variable.

Sample Audit Log The following illustrates a sample audit log for the hrnbrset utility:

```
*batchsrv      OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 19 TIME 01 28
Requested Utility: hrnbrset

HOST = REGULUS


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - OPENED


*hrnbrset            SET UP SEGMENT RTIX         Date 2007 MAR 19 Time 01:28

Table @RULESLIBRARY    -                          KEPT
Table ACCESSLOG        -                       INSERTED
Table EVENTRULES       -                          KEPT
Table FIELDS           -                          KEPT
Table ORDERING         -                          KEPT
Table PARMS            -                          KEPT
Table PROJ3278         -                       INSERTED
Table PR23936          -                       INSERTED
Table PR24936          -                       INSERTED
Table SCREENFIELDS     -                          KEPT
Table SCREENS          -                          KEPT
Table SCREENTABLES     -                          KEPT
Table SELECTION        -                          KEPT
Table TABLES           -                          KEPT


                              PAGE I/O SUMMARY

          TOTAL PAGES FREED        00000000050

          TOTAL PAGES READ         00000000029

          TOTAL PAGES UPDATED      00000000006

*          START TIME  01:28:27    END TIME    01:28:28


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - CLOSED

Connection with Host terminated
```

If an error condition exists, execution of the hrnbrset utility is terminated. You should investigate the cause of the error, correct it, and execute the hrnbrset utility again. Messages generated by this utility are explained in *TIBCO Object Service Broker Messages With Identifiers*.

hrnbrsix (Batch Secondary Index Build – TDS Tables)

Syntax hrnbrsix -C *controlfile* [*arguments*]

Platforms Windows, Solaris

Description The hrnbrsix utility provides a fast method for building one or more secondary indexes on a large TDS table. It offers significant performance improvements over the online secondary index build tool (SIXBUILD) because it can build multiple secondary indexes from just one pass through the data.

You must use this utility if your table is greater than 3000 pages or exceeds site limits for online processing.

Argument

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this argument is omitted, the report is written to *stdout*.
-b	Browse only—no data is written to the Pagestore.
-C *controlfile*	The full path and name of the control file. This entry is mandatory.
-n *buffers*	The maximum number of buffers for sorting secondary indexes. The default is four and the minimum is two.
-s *segment#*	The offline segment number. If this argument is omitted, the segment number defaults to zero (that is, the base segment is used).
-W *workpath*	The directory to be used for temporary storage. If this argument is omitted, the current directory is used.

Constraints The following constraints apply:

- The segment containing the specified table must be offline. If the segment containing the data of the table is segment 0, the Data Object Broker must be shut down.
- You must have a correctly formatted ASCII text control file describing the name, definition, and secondary index fields of the target table. The table must be completely defined. The control file can be prepared with other relevant information using the SIXBUILD_CARDS tool, or it can be created

with an ASCII editor like Notepad, vi, or a word processor in ASCII mode. For more information, refer to *TIBCO Object Service Broker Shareable Tools*.

On the first screen of the SIXBUILD_CARDS tool, there is a field called **Dynamic Block Size**, which defaults to 4096 if left blank. You can set this to 65536 for optimum performance unless you have memory constraints or maximum throughput is not required.

- Any secondary indexes to be built must not be predefined. All secondary indexes to be built must be empty; if they already exist (contain index data) they are ignored.
- The table must contain data. If the table is empty, use the online secondary index build tool, SIXBUILD. Refer to *TIBCO Object Service Broker Shareable Tools* for more information.
- You must have enough disk space to build the work files required for sorting the secondary index data prior to the build.
- The table definition is stored in the MetaStor, so the hrnbrsix utility cannot access the table definition to record the secondary indexes that have been created. Consequently, after the hrnbrsix utility runs, you must edit the table definition to make the new secondary index fields available to the Table Definer. For more information, refer to *TIBCO Object Service Broker Application Administration*.

Example

Windows

```
hrnbrsix -A audit.trail -C c:\cntl\control.file -s 1 -n 40
```

Solaris

```
hrnbrsix -A audit.trail -C /usr/cntl/control.file -s 1 -n 40
```

These command lines indicate:

- The audit report is written to the file audit.trail in the current directory.
- The control information is in file c:\cntl\control.file (Windows) or /usr1/cntl/control.file (Solaris).
- The table is located in segment 1.
- 40 x 64 KB buffers are used as the work area for sorting.

Sample Audit Log

The following illustrates a sample audit log for the hrnbrsix utility:

```
*batchsrv      OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 06 TIME 11 27
Requested Utility: hrnbrsix

CONTROL - file: Y2000SIX.CTL
TABLE  -  TDS Table: Y2000_TDS_LOAD

Fill Percentage: SIndex - 075; Group - 075
#fields - 50; #rows - 100000; #parameters - 0; #secondaries - 1
Areas: IxBuf - 4096; Ix - 14; E - 0; Parms - 0; PKeys - 4; LSIndex - 10
Sort Work: Maximum Areas - 999


FIELD NAME - KEY (PRIMARY KEY)
Definition: type= I; syn= B; len= 4; dec= 0; fld= 1; key= 1; prm= 0

FIELD NAME - FLD (SECONDARY KEY)
Definition: type= S; syn= C; len= 10; dec= 0; fld= 2; key= 0; prm= 0


**NOTE** -1 = Not Applicable.
END OF TABLE


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - OPENED


*hrnbrsix              TDS SIX BUILDER             Date 2007 MAR 06 Time 11:27

PAGESTORE SEGMENT NAME SEG01          SEGMENT # 1       SEGMENT TYPE TDS

CONTROL - file: Y2000SIX.CTL
OUTPUT -  TDS Table: Y2000_TDS_LOAD

Table's primary path begins at page: 00000084; no secondaries allocated.
 S6BBX112I TABLE HAS NO DATA


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - CONNECTION TERMINATED
```

Related Utilities

- hrnbrulb (Batch Unload (Offline))
- hrnbrulh (Batch Unload (Online))
- hrnbrtbl (Batch Load)

hrnbrtbl (Batch Load)

Syntax hrnbrtbl -C *controlfile* [*arguments*] *filename*

Platforms Windows, Solaris

Description You can use the hrnbrtbl (Batch Load) utility to:

- Load large volumes of data into a predefined table in the shortest time possible
- Accept input from a sequential file
- Support input field syntax types not normally supported by TIBCO Object Service Broker
- Load any predefined secondary indexes
- Load table instances of parameterized TDS tables

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this option is omitted, the report is written to *stdout*.
-b	Browse only—no data is written to the Pagestore.
-C *controlfile*	The full path and name of the control file. This argument is mandatory.
-m	Indicates that the input file is in z/OS Variable Block format. If the input file (*filename*) was created on Windows or Solaris using one of the offline batch unload utilities—hrnbrula (Batch Unload from Archive), hrnbrulb (Batch Unload (Offline)), hrnbrulh (Batch Unload (Online))—the -m argument must be specified on the command line. If the input file was created on z/OS and ported to Windows or Solaris, *do not* include the *-m* argument.
-n	Indicates the maximum number of buffers to be used for sorting secondary indexes. The default is 4 and the minimum is 2.
-s *segment#*	The offline segment number or segment name. If this option is omitted, the segment defaults to 0 (that is, the base segment is used).

Argument	Description
-W *workpath*	The directory to use for temporary storage. If this argument is omitted, the current directory is used.

To ensure the integrity of your data, if hrnbrtbl fails, you must run the hrnbrptr (Batch Pointer Check) utility.

Files

controlfile

A file containing the control information describing the input file and the table you want to load. You can create this file with the BATCHLOAD_CARDS tool or manually with an ASCII editor like Notepad or vi.

filename

The pathname of the file containing the data you want to load in a table.

Fixed (F) or Fixed Block (FB) format means that all records are the same length. Under TIBCO Object Service Broker for Open Systems, there is no difference between F and FB formats.

Variable Block (VB) means that each block is prefixed by a four-byte Block Descriptor Word (BDW) containing the length of the data in the block, including the four-byte BDW itself. For example, the BDW for a 256-byte block of data would appear as follows:

01	04	00	00

where the first two bytes contain the length followed by two bytes containing zeros.

Within each block, each record has a Record Descriptor Word (RDW) with exactly the same format as the BDW. If a record contained 20 bytes of data, the RDW would appear as follows:

00	18	00	00

where the first two bytes contain the length, followed by two bytes containing zeros. If this record is the first in the block, the beginning of the block would appear as follows:

01	04	00	00	00	18	00	00

followed by the first record (of 20 bytes).

Environment Variable

UNICODEDIR

Set this to indicate the directory that contains the Unicode configuration files, for example, %OS_ROOT%\database\UNICODEDIR in Windows or ${OS_ROOT}/database/UNICODEDIR in Solaris. For more information, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Constraints

Segment Offline

The segment containing the table to be loaded must be offline. This means that if the target segment for the table is segment 0, the Data Object Broker must be shut down.

Control File

You must prepare a correctly formatted ASCII control file describing the following:

- The format of your input file
- The definition of the target table
- Cross-reference information between the file and the table fields
- The name of the table to be loaded
- Other relevant control information

The file must be completely defined for the hrnbrtbl utility to determine the size of the largest record. If there are fields that you do not want loaded into the target table, define them as filler0 through filler*n* in the file definition and omit them from the table definition (if consecutive fields are not wanted, define them as one continuous field). This information can be prepared using the BATCHLOAD_CARDS tool or with an ASCII editor like Notepad, vi, or a word processor in ASCII mode. For more information, refer to *TIBCO Object Service Broker Shareable Tools*.

On the first screen of the BATCHLOAD_CARDS tool, there is a field called **Dynamic Block Size**, which defaults to 4096 if left blank. Using this screen, you can set this value only as high as 32767. However, maximum blocksize of 65536 is recommended for improved performance (especially if your table contains secondary indexes) unless you have memory constraints or maximum throughput is not required. To set the blocksize to 65536, you must edit the value in the control file after it is created by BATCHLOAD_CARDS, using an ASCII editor like Notepad or vi.

Non-parameterized Table

If the table is non-parameterized, it must be empty prior to the load.

Parameterized Table

If the table is parameterized, it can contain data provided that the table instances to be loaded do not already exist.

Primary Keys

Primary key fields must be either cross-referenced to an input field or system generated (that is, an IDgen key). They must not appear on value cards.

A table can have only one IDgen key field.

Secondary Indexes

If you want secondary indexes to be built while you load the table, you must predefine them with the online SIXBUILD tool.

Disk Space

You must have sufficient disk work space (refer to the -W argument).

EBCDIC Format

The data to be loaded must be in EBCDIC format and must be in the proper sequence (ascending—parameter 1 through *n*, primary keys—1 through *n*). If your data is in ASCII format, it must be converted to EBCDIC for the batch load to work correctly.

Validation

Some data validation is done and records are rejected if the data is bad; if the number of records rejected exceeds the reject limit, the load is aborted.

Null-equivalent Values

Numeric syntax fields (binary and packed decimal) have a null-equivalent value that is defined as the lowest possible value that the field can hold. Null values in the input file are preserved using the following default behavior:

- It is possible that binary fields do not use the following values if the null-equivalent value is used:

8 byte	-9223372036854775808 (X'8000000000000000')
4 byte	-2147483648 (X'80000000')
2 byte	-32768 (X'8000')

- Packed decimal fields use the alternative negative sign (X'B'), which means that the lowest possible value can still be used in tables. In a file where the default null-equivalent value is used, the following values must not be used:

16 byte	X'99999999999999999999999999999999B'
8 byte	X'999999999999999B'
7 byte	X'9999999999999B'
6 byte	X'99999999999B'
5 byte	X'999999999B'
4 byte	X'9999999B'
3 byte	X'99999B'
2 byte	X'999B'
1 byte	X'9B'

The default values can be overridden to use a user-defined value, the highest possible value, or no null-equivalence permitted.

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using the hrnbrulb (Batch Unload (Offline)) utility or the hrnbrulh (Batch Unload (Online)) utility, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using the hrnbrtbl utility they are loaded as a zero.

Examples

Windows

```
hrnbrtbl -A audit.log -C c:\cntl\control.file -s 1 -n 40 input.file
```

Solaris

```
hrnbrtbl -A audit.trail -C /usr1/cntl/control.file -s 1
         -n 40 input.file
```

These examples show:

- The load run details are written to the file audit.log (Windows) or audit.trail (Solaris) in the current directory.
- The control information is in the file c:\cntl\control.file (Windows) or /usr1/cntl/control.file (Solaris).
- The table is loaded to segment 1.
- 40 buffers are used to sort the secondary indexes.
- The data to be loaded is contained in the file input.file in the current directory.

Sample Audit Log

The following illustrates a sample audit log for the hrnbrtbl utility:

```
*batchsrv      OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 06 TIME 11 27
Requested Utility: hrnbrtbl

CONTROL - file: Y2000TBL.CTL
INPUT - file: TBL.IN: Data type: Huron; Record format: Variable
OUTPUT -   TDS Table: Y2000_TDS_LOAD   Character Set: ENGL

Fill Percentage: Data - 075; Index - 075; SIndex - 075; Group - 075
#fields - 50; #records - 100000; #parameters - 0; #secondaries - 0
Areas: IxBuf - 4096; Ix - 4; E - 16; Parms - 0; PKeys - 4; LSIndex - 0
Input sizes: Record - 18; Minimum Record - 8 Buffer - 65536

UNLESS SPECIFIED BELOW (COLS 64-79) THE FOLLOWING PACKED DECIMAL AND
BINARY INPUT VALUES WILL BE CONVERTED TO NULL ON THE LOADED TABLE:
BINARY: (default = LOWVALUE)  Length 2 thru 8                    x'80~~~~~~~~00'
PACKED: (default = LOWVALUE)  Length 1 thru 8    x'9B'  thru  x'999999999999999B'

LOAD FIELD: KEY ---> KEY (PRIMARY KEY)
IN:  type=  ; syn= B; len= 4; dec= 0; offset=0
OUT: type= I; syn= B; len= 4; dec= 0; fld= 1; key= 1; prm= 0

LOAD FIELD: FLD ---> FLD
IN:  type=  ; syn= C; len= 10; dec= 0; offset=4
OUT: type= S; syn= C; len= 10; dec= 0; fld= 2; key= 0; prm= 0

**NOTE** -1 = Not Applicable.
END OF TABLE


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - OPENED

*hrnbrtbl              TDS TABLE LOAD              Date 2007 MAR 06 Time 11:27

PAGESTORE SEGMENT NAME SEG01          SEGMENT # 1       SEGMENT TYPE TDS

CONTROL - file: Y2000TBL.CTL
INPUT - file: TBL.IN: Data type: Huron; Record format: Variable
OUTPUT -     TDS Table: Y2000_TDS_LOAD   Character Set: ENGL

Table's primary path begins at page: 00000084; no secondaries allocated.

 S6BBL148E NO INPUT RECORDS READ, PROCESSING TERMINATED

PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - CONNECTION TERMINATED
```

Related Utilities

- hrnbrulb (Batch Unload (Offline))
- hrnbrulh (Batch Unload (Online))
- hrnbrsix (Batch Secondary Index Build – TDS Tables)

hrnbrula (Batch Unload from Archive)

Syntax hrnbrula -C *controlfile* [*arguments*] *archivefile*

Platforms Windows, Solaris

Description the hrnbrula utility is a TIBCO Object Service Broker recovery utility that restores one or more TDS tables from an archive file. You can:

- Recover TDS data from an archive file in unload format
- Select and unload table instances from parameterized tables or entire tables

Phases of hrnbrula Processing

There are three phases to the unload table from archive process. During the process, the archive file is scanned twice. The utility accepts a number of arguments to control both the program flow and storage utilization. The three phases of the table unload process are:

Phase	Process
1	Scans the archive file and extracts information from selected page types. At the end-of-file, the archive checks to ensure that a root page exists for each requested entry.
2	Identifies which data pages contain the information for all the unloaded tables.
3	Extracts the data occurrences from the required pages and produces a work file that is sorted into appropriate order based on parameter values within the table name. The output from the third phase is the RECOVER file.

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. This entry is optional. If omitted, the report is written to the default file *stdout*.
-C *controlfile*	The full path and name of the control file. This entry is mandatory.
-d *datapage%*	Data page %. This number indicates the approximate % of used pages that are D pages. The entry is optional and has a default of 85%.
-g *group%*	Group index page %. Approximate % of used segment pages that are G pages. This is an optional entry and has a default of 10%.
-h *groupindex%*	Group-index index page %. Approximate % of used segment pages that are H pages. This is an optional entry and has a default of 2%.
-i *dataindex%*	Data Index page %. Approximate % of used segment pages that are I pages. This is an optional entry and has a default of 3%.
-M *3-char-prefix*	This entry indicates a separate file is required for each table or instance. You must supply either an -M *3 character prefix* or a -U *unloadfile* argument.
-n *buffer#*	Number of sort buffers to be used. The entry is optional and has a default of 10 buffers.
-p *pages#*	Approximate number of used pages in segment. This optional entry has a default of 100000. In cases where this number is unknown, enter the total number of pages in the segment. The default number is usually sufficient. If not, it is better to overestimate.
-S *selectionfile*	Full path of selection control file. This entry is required only if selected parameter instances of a table are being recovered.
-s *segment#*	The offline segment number. This argument defaults to 0 (that is, the base segment is used).
-U *unloadfile*	The full path and name of the unload file. You must supply either a -U *unloadfile* or an -M *3 character prefix* argument.

Argument	Description
-v *parameter%*	Parameter value %. Approximate % of G pages participating in the recovery. This optional entry has a default of 1%.
-W *workdir*	The full path of the work directory. This entry is optional and defaults to the current directory.

When in doubt, allow % arguments to default, as the defaults are sufficient to handle most unload situations.

Files

controlfile

A file containing the control information describing the output file and the table to unload. Create this file using the BATCHUNLD_CARDS tool.

archivefile

The name of the archive file.

Environment Variable

UNICODEDIR

Set this to indicate the directory that contains the Unicode configuration files, for example, %OS_ROOT%\database\UNICODEDIR in Windows or ${OS_ROOT}/database/UNICODEDIR in Solaris. For more information, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Constraints

Control File

You must have a correctly formatted control file, describing the format of your table (which is also the format of your output file), the table name to be unloaded, and other relevant control information. This information can be prepared using the BATCHUNLD_CARDS tool or with an ASCII editor like Notepad or vi. For more information, refer to *TIBCO Object Service Broker Shareable Tools*.

The complete file must be defined so that the hrnbrula utility can determine the size of the maximum record.

Selection File

If the table to be unloaded is parameterized and you want to unload only selected instances, you must have a correctly formatted selection file (ASCII text) constructed using the online BATCHUNLD_CARDS tool. This selection file indicates which parameter instances you want to include or exclude.

Parameterized Table Instances

Tables must not be empty. Parameterized table instances selected for inclusion in the unloaded file by using the equals (=) relational operator must exist in the table.

Reloading the Data

If you use one of the Batch Unload utilities to unload data, reload the data by using the Batch Load utility. You cannot reload the data using the interactive LOAD tool.

EBCDIC Format

The output file is created in EBCDIC (z/OS variable) format.

Variable Block (VB) means that each block is prefixed by a four-byte Block Descriptor Word (BDW) containing the length of the data in the block, including the four-byte BDW itself. For example, the BDW for a 256-byte block of data would appear as follows:

01	04	00	00

where the first two bytes contain the length followed by two bytes containing zeros.

Within each block, each record has a Record Descriptor Word (RDW) with exactly the same format as the BDW. If a record contained 20 bytes of data, the RDW would appear as follows:

00	18	00	00

where the first two bytes contain the length, followed by two bytes containing zeros. If this record is the first in the block, the beginning of the block would appear as follows:

01	04	00	00	00	18	00	00

followed by the first record (of 20 bytes).

Null-equivalent Values

Numeric syntax fields (binary and packed decimal) have a null-equivalent value, which is defined as the lowest possible value that the field can hold. Null values in the input file are preserved using the following default behavior:

- It is possible that binary fields do not use the following values if the null-equivalent value is used:

8 byte	-9223372036854775808 (X'8000000000000000')
4 byte	-2147483648 (X'80000000')
2 byte	-32768 (X'8000')

- Packed decimal fields use the alternative negative sign (X'B'), which means that the lowest possible value can still be used in tables. In a file where the default null-equivalent value is used, the following values must not be used:

16 byte	X'99999999999999999999999999999999B'
8 byte	X'999999999999999B'
7 byte	X'99999999999999B'
6 byte	X'9999999999B'
5 byte	X'999999999B'
4 byte	X'99999999B'
3 byte	X'99999B'
2 byte	X'999B'
1 byte	X'9B'

The default values can be overridden to use a user-defined value, the highest possible value, or no null-equivalence permitted.

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using the hrnbrula utility, the hrnbrulb (Batch Unload (Offline)) utility, or the hrnbrulh (Batch Unload (Online)) utility, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using the hrnbrtbl (Batch Load) utility, they are loaded as a zero.

Using FTP to Transfer Unloaded Data

If you are using FTP to transfer unloaded data between z/OS and Windows or Solaris, you must reformat the unloaded data using the S6BBRFRU z/OS utility before it can be used by TIBCO Object Service Broker. Refer to *TIBCO Object Service Broker for z/OS Utilities* for more information on the S6BBRFRU utility.

Examples The following examples illustrate the use of the hrnbrula utility:

Windows

```
hrnbrula -A audit.report -C c:\cntl\control.file
         -S c:\brula\select.file -s 10 -n 40 -W c:\tmp
         -M c:\brula\TBL c:\archives\seg10.arc
```

Solaris

```
hrnbrula -A audit.report -C /usr1/cntl/control.file
         -S /usr1/brula/select.file -s 10 -n 40 -W /tmp
         -M /usr1/brula/TBL usr1/archives/seg10.arc
```

This produces an unload with a separate file for each table where:

- Unloaded tables are written to files TBL001 through TBLnnn in the directory c:\brula (Windows) or /usr1/brula (Solaris). *TBL* is the 3 character prefix designated by the -M argument.
- The audit report is created in the current directory with the filename audit.report.
- Unload uses the control file called control.file in the directory c:\cntl (Windows) or /usr1/cntl (Solaris).
- Unload uses the select file called select.file in the directory c:\brula (Windows) or /usr1/brula (Solaris).
- Archive is a backup of segment 10.
- Sort can use a maximum of 40 x 64 KB buffers.
- Work disk space is available in c:\tmp (Windows) or /tmp (Solaris).
- Archive used as input is located in the directory c:\archives (Windows) or /usr1/archives (Solaris).

Sample Audit Log The following is a compressed version of a sample audit log for the hrnbrula utility:

```
*hrnbrula   TABLE RECOVERY ARCHIVE EXTRACT SEG#  001   DATE 2007 MAR 06 TIME 02:18

From Archive: seg01.bak
Control File: Y2000ULA.CTL
No Selection
Output File: ULA.OUT
Use current directory for work-file space
# of Tables requested: 1
No Parameterized tables included
BREAK RUN option not requested

REQUIREMENT TYPE   % OF PAGES  PAGE ESTIMATES     MEMORY REQUIRED

DATA                       85           85000             1700000
PRIMARY INDEX               3            3000               24000
GROUP INDEX                10           10000              200000
GROUP INDEX INDEX           2            2000               16000
TOTAL PAGES               100          100000             1940000

PARM INSTANCE INDEX         1             100              810012
TOTAL MEMORY      ------------------------------          2750012

-------------------- TABLE # 0001 -----------------------
CONTROL - file: Y2000ULA.CTL
INPUT  -    TDS Table: Y2000_TDS   Character Set: ENGL
OUTPUT - file: : Data type: Huron; Record format: Variable

#fields - 2; #records - 100000; #parameters - 0; #secondaries - 0
Output sizes: Record - 18; Minimum Record - 3 Buffer - 65536

*************
S6BBA011I PASS ONE PROCESSING INITIATED

 S6BBA012I PASS ONE:   READ
PAGES:        HIX         GIX          IX        DATA      OTHERS
                0           5           1         149          32

S6BBA013I PASS TWO PROCESSING INITIATED

 S6BBA015I DATA TABLE COMPRESSION COMPLETE
------> Input: 149    Output: 1
Input: 0  Processed: 0  - S6BBA019I PASS TWO; GROUP INDEX PAGES READ


S6BBA020I PASS THREE PROCESSING INITIATED

RECOVERING TABLE: Y2000_TDS   S6BBA159I TABLE OR INSTANCE EMPTY
```

```
Input: 188  Processed: 1  - S6BBA021I PASS THREE; ARCHIVE RECORDS

Y2000_TDS --->  S6BBA159I TABLE OR INSTANCE EMPTY


----------------------------------------
SORT INITIALIZATION: Work area = 262008
----------------------------------------

SORTING COMPLETED - FINAL PHASE INITIATED:

FILE:    ULA.OUT :
  S6BSV997I MVS Dataset characteristics:
  ----> RECFM=VB,LRECL=38,BLKSIZE=42
Y2000_TDS           # rows unloaded:  1

 S6BBA160I **AT LEAST 1 TABLE OR INSTANCE EMPTY**
```

Related Utilities

- hrnbrulb (Batch Unload (Offline))
- hrnbrulh (Batch Unload (Online))
- hrnbrtbl (Batch Load)
- hrnbrsix (Batch Secondary Index Build – TDS Tables)

hrnbrulb (Batch Unload (Offline))

Syntax hrnbrulb -C *controlfile* [*arguments*] *filename*

Platforms Windows, Solaris

Description If TIBCO Object Service Broker is shut down or the Pagestore segment containing the table is offline, use the hrnbrulb utility to unload data. This utility provides you with:

- A means of unloading large volumes of data from TDS tables
- An unload capability on a system that is offline
- Table instance selection within parameterized table sets
- Data in a readily usable format (a sequential flat file)

The two unload utilities (hrnbrulb and hrnbrulh (Batch Unload (Online))) are similar, and differ only in the following ways:

- The hrnbrulb utility is an *offline* unload, while the hrnbrulh utility is an *online* unload.
- In the hrnbrulb utility, you must specify the segment in which your table resides. In the hrnbrulh (Batch Unload (Online)) utility, you must identify the Data Object Broker to be accessed.
- While the hrnbrulb utility provides a committed table unload, the hrnbrulh (Batch Unload (Online)) utility does not necessarily do so, depending on concurrent online activity.

Arguments

Argument	Description
-A *auditreport*	The file to which the audit report is written. If this argument is omitted, the report is written to the screen (Windows) or *stdout* (Solaris).
-C *controlfile*	The full path and name of the control file that is constructed using the BATCHUNLD_CARDS shareable tool. This entry is mandatory.
-s *segment#*	The offline segment number. If this argument is omitted, the segment number defaults to 0 (that is, the base segment is used).
-S *selectionfile*	The full path and filename of the selection file that is constructed using the BATCHUNLD_CARDS shareable tool. This entry is optional.

Argument	Description
-O *unloadfile*	Indicates oldnull argument is selected. All numeric nulls encountered in the unload file are treated as zero.

Files

controlfile

A file containing the control information describing the output file and the table to unload. Create this file using the BATCHUNLD_CARDS tool.

filename

The name of the file to which the table is unloaded.

Environment Variable

UNICODEDIR

Set this to indicate the directory that contains the Unicode configuration files, for example, %OS_ROOT%\database\UNICODEDIR in Windows or ${OS_ROOT}/database/UNICODEDIR in Solaris. For more information, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Constraints

Segment Offline

The Pagestore segment from which you intend to unload a table must be offline. This means that if the segment containing the table's data is segment 0, the Data Object Broker must be shut down.

Control File

You must have a correctly formatted control file, describing the format of your table (which is also the format of your output file), the table name to be unloaded, and other relevant control information. This file is prepared using the BATCHUNLD_CARDS shareable tool and supplied to the utility using the -C argument. For more information, refer to *TIBCO Object Service Broker Shareable Tools*.

The complete file must be defined so that the hrnbrulb utility can determine the size of the maximum record.

Selection File

If the table to be unloaded is parameterized and you want to unload only selected instances, you must have a correctly formatted selection file (ASCII text) constructed using the online BATCHUNLD_CARDS tool and supplied to the utility using the -S argument. For more information, refer to *TIBCO Object Service Broker Shareable Tools*. This selection file indicates which parameter instances you want to include or exclude.

Parameterized Table Instances

Tables must not be empty. Parameterized table instances selected for inclusion in the unloaded file by using the equals (=) relational operator must exist in the table.

Reloading the Data

If you use either of the Batch Unload utilities to unload data, reload the data by using the Batch Load utility. You cannot reload the data using the interactive LOAD tool.

EBCDIC Format

The output file is created in EBCDIC (z/OS variable) format.

Fixed (F) or Fixed Block (FB) format means that all records are the same length. Under OSB for Windows or OSB for Solaris, there is no difference between F and FB formats.

Variable Block (VB) means that each block is prefixed by a four-byte Block Descriptor Word (BDW) containing the length of the data in the block, including the four-byte BDW itself. For example, the BDW for a 256-byte block of data would appear as follows:

01	04	00	00

where the first two bytes contain the length followed by two bytes containing zeros.

Within each block, each record has a Record Descriptor Word (RDW) with exactly the same format as the BDW. If a record contained 20 bytes of data, the RDW would appear as follows:

00	18	00	00

where the first two bytes contain the length, followed by two bytes containing zeros. If this record is the first in the block, the beginning of the block would appear as follows:

01	04	00	00	00	18	00	00

followed by the first record (of 20 bytes).

Null-equivalent Values

Numeric syntax fields (binary and packed decimal) have a null-equivalent value that is defined as the lowest possible value that the field can hold. Null values in the input file are preserved using the following default behavior:

- It is possible that binary fields do not use the following values if the null-equivalent value is used:

8 byte	-9223372036854775808 (X'8000000000000000')
4 byte	-2147483648 (X'80000000')
2 byte	-32768 (X'8000')

- Packed Decimal fields use the alternative negative sign (X'B'), which means that the lowest possible value can still be used in tables. In a file where the default null-equivalent value is used, the following values must not be used:

16 byte	X'99999999999999999999999999999999B'
8 byte	X'999999999999999B'
7 byte	X'9999999999999B'
6 byte	X'99999999999B'
5 byte	X'999999999B'
4 byte	X'9999999B'
3 byte	X'99999B'
2 byte	X'999B'
1 byte	X'9B'

The default values can be overridden to use a user-defined value, the highest possible value, or no null-equivalence permitted.

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using the hrnbrulb utility or the hrnbrulh (Batch Unload (Online)) utility, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using the hrnbrtbl (Batch Load) utility, they are loaded as a zero.

Transferring Unloaded Data Using FTP

If you are using FTP to transfer unloaded data between z/OS and Windows or Solaris, you must reformat the unloaded data using the S6BBRFRU z/OS utility before it can be used by TIBCO Object Service Broker. Refer to *TIBCO Object Service Broker for z/OS Utilities* for more information on the S6BBRFRU utility.

Examples

Windows

```
hrnbrulb -A audit.trail -C c:\cntl\control.file -s 1
         -O output.file
```

Solaris

```
hrnbrulb -A audit.trail -C /usr1/cntl/scontrol.file -s 1
         -O output.file
```

These command lines indicate:

- The unload details are written to the file audit.trail in the current directory.
- The control information is in the file c:\cntl\control.file (Windows) or /usr1/cntl/control.file (Solaris).
- The table is to be unloaded from segment 1.
- All numeric nulls encountered in the unload file are treated as zero.
- The unload file is output.file in the current directory.

Sample Audit Log

The following illustrates a sample audit log for the hrnbrulb utility:

```
*batchsrv      OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 06 TIME 11 27
Requested Utility: hrnbrulb

CONTROL - file: Y2000ULB.CTL
INPUT  -  TDS Table: Y2000_TDS   Character Set: ENGL
OUTPUT - file: ULB.OUT: Data type: Huron; Record format: Variable

#fields - 2; #records - 100000; #parameters - 0; #secondaries - 0
Output sizes: Record - 18; Minimum Record - 3 Buffer - 65536
```

```
UNLESS SPECIFIED BELOW (COLS 64-79) PACKED DECIMAL AND BINARY NULLS WILL
BE CONVERTED AND OUTPUT TO THE FILE AS FOLLOWS:
BINARY: (default = LOWVALUE)  Length 2 thru 8                    x'80~~~~~~~~00'
PACKED: (default = LOWVALUE)  Length 1 thru 8    x'9B'  thru  x'999999999999999B'


UNLOAD FIELD - KEY: type= I; syntax= B; length= 4; decimals= 0

UNLOAD FIELD - FLD: type= S; syntax= C; length= 10; decimals= 0


END OF TABLE

PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - OPENED


*hrnbrulb         TDS TABLE UNLOAD            Date 2007 MAR 06 Time 11:27

Table's primary path begins at page: 0100004F; no secondaries allocated.

  S6BBU100E: TABLE TO BE UNLOADED DOES NOT CONTAIN ANY DATA


PAGESTORE SEGMENT NAME SEG01   SEGMENT # 1  TYPE TDS - CONNECTION TERMINATED
```

Related Utilities

- hrnbrulh (Batch Unload (Online))
- hrnbrtbl (Batch Load)
- hrnbrsix (Batch Secondary Index Build – TDS Tables)

hrnbrulh (Batch Unload (Online))

Syntax

hrnbrulh -C *controlfile* [*arguments*] *filename*

Platforms

Windows, Solaris

Description

The hrnbrulh utility extracts data from an entire table or from selected table instances of a parameterized table. It is used when TIBCO Object Service Broker is up and running and the Pagestore segment containing the data to be unloaded is online. This unload utility provides you with:

- A means of unloading large volumes of data from TDS tables
- An unload capability on an active system
- Table instance selection within parameterized table sets
- Data in a readily usable format (a sequential flat file)

Unload Utilities Differences

The two unload utilities (hrnbrulh and hrnbrulb (Batch Unload (Offline)) are similar, and differ only in the following ways:

Online/Offline

The hrnbrulh utility is an *online* unload, while the hrnbrulb utility is an *offline* unload.

Specifying Table Location

In the hrnbrulh utility, you identify the Data Object Broker to be accessed. In the hrnbrulb utility, you specify the segment where your table resides.

Security on the hrnbrulh Utility

Unload Restrictions

Unloading is *not* permitted under either of the following conditions:

- The target table is a restricted system table
- The target table has protected table instances (parameters)

Permissions Enforced

The following permissions are enforced for allowable unloads:

- If the requestor is either the owner of the target table or the security administrator of the owner, the request is allowed.

  If the requestor is neither the owner of the target table nor the security administrator of the owner, and the requestor's security classification is less than the security clearance required of the target table, the request is denied.
- If the requestor has READ ACCESS to the target table by virtue of direct permission granted to the user ID or the current group, for example, the request is allowed; otherwise, the request is denied.

Arguments

Argument	Description
-A *auditreportname*	The file to which the audit report is written. If this argument is omitted, the report is written to the screen (Windows), or *stdout* (Solaris).
-C *controlfile*	The full path and name of the control file that is constructed using the BATCHUNLD_CARDS shareable tool. This entry is mandatory.
-O	Indicates oldnull argument is selected. All numeric nulls encountered in the unload file are treated as zero.
-S *selectionfile*	The full path and filename of the selection file that is constructed using the BATCHUNLD_CARDS shareable tool. This entry is optional.
-t *nodename*	The name of the Data Object Broker through which the table is unloaded.
-U *userid*	The user ID used to determine authorities. This argument must always be used in conjunction with the -P argument. If not specified, the login user ID is the default.
-P *password*	The password for the user ID specified using the -U argument.

Files

controlfile

A file containing the control information describing the output file and the table you want to unload. Create this file using the BATCHUNLD_CARDS tool.

filename

The name of the file to which the table is unloaded.

Environment Variable

UNICODEDIR

Set this to indicate the directory that contains the Unicode configuration files, for example, %OS_ROOT%\database\UNICODEDIR in Windows or ${OS_ROOT}/database/UNICODEDIR in Solaris. For more information, refer to *TIBCO Object Service Broker for Open Systems Installing and Operating*.

Constraints

Segment Online

The Pagestore segment from which you intend to unload a table must be online. This means that the Data Object Broker must be in operation.

Control File

You must have a correctly formatted Control File, describing the format of your table (which is also the format of your output file), the table name to be unloaded, and other relevant control information. This file is prepared using the BATCHUNLD_CARDS shareable tool and supplied to the utility using the -C argument. For more information, refer to *TIBCO Object Service Broker Shareable Tools*.

The complete file must be defined so that the hrnbrulh utility can determine the size of the maximum record.

Selection File

If the table to be unloaded is parameterized and you want to unload only selected instances, you must have a correctly formatted selection file (ASCII text) constructed using BATCHUNLD_CARDS and supplied to the utility using the -S argument. For more information, refer to *TIBCO Object Service Broker Shareable Tools*. This selection file indicates which parameter instances you want to include or exclude.

Parameterized Table Instances

Tables must not be empty. Parameterized table instances selected for inclusion in the unloaded file by using the equals (=) relational operator must exist in the table.

Reloading the Data

If you use either of the Batch Unload utilities to unload data, you can reload the data by using the Batch Load utility. You cannot reload the data using the interactive LOAD tool.

EBCDIC Format

The output file is created in EBCDIC (z/OS variable) format only.

Fixed (F) or Fixed Block (FB) format means that all records are the same length. Under OSB for Windows or OSB for Solaris, there is no difference between F and FB formats.

Variable Block (VB) means that each block is prefixed by a four-byte Block Descriptor Word (BDW) containing the length of the data in the block, including the four-byte BDW. For example, the BDW for a 256-byte block of data would appear as follows:

01	04	00	00

where the first two bytes contain the length followed by two bytes containing zeros.

Within each block, each record has a Record Descriptor Word (RDW) with exactly the same format as the BDW. If a record contained 20 bytes of data, the RDW would appear as follows:

00	18	00	00

where the first two bytes contain the length, followed by two bytes containing zeros. If this record is the first in the block, the beginning of the block would appear as follows:

01	04	00	00	00	18	00	00

followed by the first record (of 20 bytes).

Null-equivalent Values

Numeric syntax fields (binary and packed decimal) have a null-equivalent value that is defined as the lowest possible value that the field can hold. Null values in the input file are preserved using the following default behavior:

- It is possible that binary fields do not use these values if the null-equivalent value is used:

8 byte	-9223372036854775808 (X'8000000000000000')
4 byte	-2147483648 (X'80000000')
2 byte	-32768 (X'8000')

- Packed Decimal fields use the alternative negative sign (X'B'), which means that the lowest possible value can still be used in tables. In a file where the default null-equivalent value is used, the following values must not be used:

16 byte	X'99999999999999999999999999999999B'
8 byte	X'999999999999999B'
7 byte	X'99999999999999B'
6 byte	X'9999999999B'
5 byte	X'999999999B'
4 byte	X'99999999B'
3 byte	X'99999B'
2 byte	X'999B'
1 byte	X'9B'

The default values can be overridden to use a user-defined value, the highest possible value, or no null-equivalence permitted.

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using the hrnbrulb (Batch Unload (Offline)) utility or the hrnbrulh utility, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using the hrnbrtbl (Batch Load) utility, they are loaded as a zero.

Transferring Unloaded Data Using FTP

If you are using FTP to transfer unloaded data between z/OS and Windows or Solaris, you must reformat the unloaded data using the S6BBRFRU z/OS utility before it can be used by TIBCO Object Service Broker. Refer to *TIBCO Object Service Broker for z/OS Utilities* for more information on the S6BBRFRU utility.

Examples

The following are examples of the hrnbrulh command:

Windows

```
hrnbrulh -A audit.trl -C c:\cntl\control.fil -O
         -t dbid output.fil
```

Solaris

```
hrnbrulh -A audit.trl -C /usr1/cntl/control.fil -O
         -t dbid output.fil
```

These command lines indicate:

- The unload details are written to the file audit.trl in the current directory.
- The control information is in c:\cntl\control.fil (Windows) or /usr1/cntl/control.fil (Solaris).
- All numeric nulls encountered in the unload file are treated as zero.
- The table is to be unloaded from the Data Object Broker *dbid*.
- The unload file is output.fil in the current directory.

Sample Audit Log

The following illustrates a sample audit log for the hrnbrulh utility:

```
*batchsrv     OFFLINE BATCH UTILITY SERVER         DATE 2007 MAR 06 TIME 11 27
Requested Utility: hrnbrulh

DEFAULT HOST, no (-t) HOST-ID submitted
HOST = TEST
CONTROL - file: Y2000ULH.CTL
INPUT  - TDS Table: Y2000_TDS   Character Set: ENGL
OUTPUT - file: ULH.OUT: Data type: Huron; Record format: Variable

#fields - 2; #records - 100000; #parameters - 0; #secondaries - 0
Output sizes: Record - 18; Minimum Record - 3 Buffer - 65536


UNLESS SPECIFIED BELOW (COLS 64-79) PACKED DECIMAL AND BINARY NULLS WILL
BE CONVERTED AND OUTPUT TO THE FILE AS FOLLOWS:
BINARY: (default = LOWVALUE)  Length 2 thru 8                    x'80~~~~~~~~00'
PACKED: (default = LOWVALUE)  Length 1 thru 8    x'9B'  thru  x'999999999999999B'


UNLOAD FIELD - KEY: type= I; syntax= B; length= 4; decimals= 0

UNLOAD FIELD - FLD: type= S; syntax= C; length= 10; decimals= 0


END OF TABLE



*hrnbrulh             TDS TABLE UNLOAD            Date 2007 MAR 06 Time 11:27

usr10 UNLOADING TABLE Y2000_TDS FOR SYSADMIN

Get page request to Host failed
RTIX Index not available  -  S6BBU106E: RESIDENT TABLE INDEX ENTRY NOT FOUND FOR
TABLE

Connection with Host terminated
```

Related Utilities

- hrnbrulb (Batch Unload (Offline))
- hrnbrtbl (Batch Load)
- hrnbrsix (Batch Secondary Index Build – TDS Tables)

hrncr (TIBCO Object Service Broker Data Object Broker)

Syntax hrncr [*arguments*] [*subcommand*]

Platforms Windows, Solaris

Description This command is used to start, maintain, and stop the Data Object Broker. By default it starts the Data Object Broker, and further subcommands can be used to operate and shut down the Data Object Broker.

See Also *TIBCO Object Service Broker for Open Systems Installing and Operating* for more information on subcommands used with the hrncr utility.

Arguments

Argument	Description
-c *hurondir*	Overrides the path to the TIBCO Object Service Broker *huron.dir* file. By default, this file is in the directory pointed to by the *HURONDIR* environment variable, if it exists, or in the TIBCO Object Service Broker database directory.
-d *databasedir*	Overrides the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-f	Allows forking so that a new window appears for the Data Object Broker supervisor. This window remains open for as long as the Data Object Broker is running.
-h *huronpathdir*	Overrides the TIBCO Object Service Broker installation directory. By default this is extracted from the *OS_ROOT* environment variable. If the OS_ROOT variable is not set, this argument must be specified.
-l *logdir*	Overrides the TIBCO Object Service Broker log directory. By default this is the log directory under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.

Argument	Description
-o *parmoverrides*	Overrides one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`
-p *parmfile*	Overrides the location of the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).

Subcommands

The following subcommands are available:

Subcommand	Description
?	Display command usage information.
canceluser=*userid* **canceluser**=*userid;terminalid* **canceluser**=*userid;commnum*	Disconnects a user ID from the Data Object Broker specified by *dob_jobname*. An optional terminal ID may be supplied if the same user is logged in more than once. If DUPUSERID is set to Y, and you enter just a *user ID*, you are prompted to add a unique connection identifier (*commnum*) to the command.
checkpoint	Force an immediate checkpoint.
dboffline=*segname* or *segnumber*	Move the specified segment offline.
dbonline=*segname* or *segnumber*	Move the specified segment online.
kill	Stop the Data Object Broker without the normal shutdown procedure. This subcommand should be used only if the Data Object Broker is not responding to a shutdown command or if there is an urgent need to bring it down quickly.

Subcommand	Description
log	Monitor the active Data Object Broker log file: • Windows: A new window is created to display the log contents. • Solaris: The last several lines of the Data Object Broker log are copied to *stdout* as well as all subsequent log entries. This subcommand must be terminated by the Break key.
parm=*parmname*=*value*	Dynamically modify a Data Object Broker parameter.
resume	Cancel a quiescent state, if set.
shutdown or **stop**	Shut down the Data Object Broker in an orderly fashion. The request is sent to the active Data Object Broker. The shutdown procedure can take anywhere from half a minute to a couple of minutes, depending on what activity had been going on shortly before shutdown.
spinsubmit=i | d	Submit a journal spin job, either immediately (i) or deferred until the next checkpoint (d).
state	The state of the Data Object Broker is written to the screen (Windows) or *stdout* (Solaris). If the Data Object Broker is running, the exit code from this subcommand is 0; otherwise it is 1.

Subcommand	Description
stopserver=alldbms | allhuron | all*servertype* | allremote | svrid *serverid* | *serveruserid*	Disconnect one or more external servers. Specify a *connection ID*, a *serverid*, or use the **all** argument. Its syntax is: **alldbms** Stop all external database servers **allhuron** Stop all inbound connections **allservertype** Stop all servers of type *servertype*, for example, allprs stops all peer servers **allremote** Combines alldbms and allprs **svrid serverid** Stop all external servers with ID *serverid* **serveruserid** Stop all servers with user ID of serveruserid The *connection ID* and *serverid* arguments can use the wildcard asterisk (*) as a suffix. For example, `serverid=srv*` means include all IDs that have the prefix "srv".
traceid=*userid*	Traces all Data Object Broker service activity for the given user ID. You can specify a full system service trace using the value *sstrace*.

Environment Variables

OS_ROOT

Set this to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utilities

- hrncrSvc (Install Data Object Broker as a Windows Service)
- hrntladm (Administration Menu)

hrncrSvc (Install Data Object Broker as a Windows Service)

Syntax hrncrSvc [*arguments*]

Platform Windows

Description This command is used to install the Data Object Broker as a Windows service, as well as update the appropriate Windows Registry entries. After installation, the Data Object Broker service has a startup type of "Manual". You can change this option to "Automatic" so that the service starts each time you start Windows.

See Also *TIBCO Object Service Broker for Open Systems Installing and Operating* for more information on operating the Data Object Broker as a Windows service.

Arguments

Argument	Description
-c *hurondir*	Overrides the path to the TIBCO Object Service Broker *huron.dir* file. By default, this file is in the directory pointed to by the HURONDIR environment variable, if it exists, or in the TIBCO Object Service Broker database directory.
-d *databasedir*	Overrides the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (OS_ROOT\database).
-h *huronpathdir*	Overrides the TIBCO Object Service Broker installation directory. By default this is extracted from the *OS_ROOT* environment variable. If the *OS_ROOT* variable is not set, this argument must be specified.
-i *svcname*	Installs the Data Object Broker as a Windows service, with the name specified, and creates the required Windows Registry entries. The default for the name is "DataObjectBroker". You can install multiple Data Object Brokers by running hrncrSvc more than once, specifying unique names each time.
-l *logdir*	Overrides the TIBCO Object Service Broker log directory. By default this is the log directory under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log). This argument is ignored if the Data Object Broker is active.

Argument	Description
-o *parmoverrides*	Overrides one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`
-p *parmfile*	Overrides the location of the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm).
-u *svcname*	Uninstalls the Data Object Broker service specified and removes the associated Windows Registry entries.
-v *svcname*	Displays whether the Data Object Broker service specified is installed.

See Also

The hrncr (TIBCO Object Service Broker Data Object Broker) utility for information about environment variables and files.

Related Utility

hrncr (TIBCO Object Service Broker Data Object Broker)

hrnspjex (Journal Extraction (or Journal Spin))

Syntax hrnspjex [*arguments*] *jrnlno spinfile*

Platforms Windows, Solaris

Description This utility is part of the TIBCO Object Service Broker backup and recovery process. The hrnspjex utility extracts the contents of a journal (JOURNAL1 or JOURNAL2) and copies the contents to a spin file that can be merged with a segment backup. The journal must not be in use at the time.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* for information about TIBCO Object Service Broker backup and recovery.

Arguments

Argument	Description
-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\database in Windows, ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is extracted from the environment variable *OS_ROOT*. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory, %OS_ROOT%\log (Windows) or ${OS_ROOT}/log (Solaris). This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is %OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).

Argument	Description
jrnlno	Specify 1 or 2 to represent JOURNAL1 or JOURNAL2.
spinfile	The file to which the offloaded data is written.
-w	Overwrite *spinfile* if it already exists.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location of %OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris.

Files

For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utilities

- hrntlbps (Backup Pagestore)
- hrntlrps (Restore Pagestore)
- hrntlmrg (Journal Merge)
- hrnspset (Reset Journal)

hrnspset (Reset Journal)

Syntax hrnspset [*arguments*] *jrnlno*

Platforms Windows, Solaris

Description This command resets a full journal (JOURNAL1 or JOURNAL2) so that it can be reused by TIBCO Object Service Broker. The journal must not be in use at the time. This utility is part of the TIBCO Object Service Broker backup and recovery process.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the installation directory (%OS_ROOT%\database in Windows, ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the installation directory. By default this is deduced from the *OS_ROOT* environment variable. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the default TIBCO Object Service Broker log directory (%OS_ROOT%\log in Windows, ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris).
jrnlno	Specify 1 or 2 to represent JOURNAL1 or JOURNAL2.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location(%OS_ROOT%\database\huron.dir in Windows, ${OS_ROOT}/database/huron.dir in Solaris).

Files

For a full description of the following files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utility

hrnspjex (Journal Extraction (or Journal Spin))

hrntladm (Administration Menu)

Syntax hrntladm [*arguments*]

Platforms Windows, Solaris

Description hrntladm is an administration utility that helps you monitor and control the TIBCO Object Service Broker environment. Using this menu-based utility, you can display statistics, generate diagnostic dumps, and browse Data Object Broker data.

See Also *TIBCO Object Service Broker for Open Systems Installing and Operating* for a description of all arguments on the Administration menu.

Arguments

Argument	Description
-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is deduced from the *OS_ROOT* environment variable. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is %OS_ROOT%\database\crparm in Windows and ${OS_ROOT}/database/crparm in Solaris.

Argument	Description
-t *nodename*	Administer the Data Object Broker identified by *nodename*.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris)

Files

For a full description of the following files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utility

hrncr (TIBCO Object Service Broker Data Object Broker)

hrntlbps (Backup Pagestore)

Syntax hrntlbps [*arguments*] *backupfile*

Platforms Windows, Solaris

Description The hrntlbps utility can be called as part of the TIBCO Object Service Broker backup process. This command is used to back up a TDS segment. It takes all the used pages in a segment, strips unused space from them, and writes them to the backup file. The segment must be offline. If the -s argument is not specified, the hrntlbps utility backs up segment 0 (the MetaStor) by default.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-a	Append the segment data to the backup file.
-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is extracted from the *OS_ROOT* environment variable. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`

Argument	Description
-p *parmfile*	Override the Data Object Broker parameter file. By default this is the file crparm under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).
-s *segname* or *segnumber*	Back up the segment named *segname* or *segnumber*.
-v	Checks that the segment number in each page backed up is the same as the segment number requested. If the segment is not a true TIBCO Object Service Broker segment, the backup fails when it encounters the incorrect segment number.
-w	Overwrite the backup file if it exists.

Environment Variables

OS_ROOT

Set to the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files

For details, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utilities

- hrntlrps (Restore Pagestore)
- hrnspjex (Journal Extraction (or Journal Spin))
- hrntlmrg (Journal Merge)

hrntlfcl (Format Contingency Log)

Syntax hrntlfcl [*arguments*] *contingencylog*

Platforms Windows, Solaris

Description This command is used to format the contingency log. The contingency log must be identified with its full path name, as in the following examples:

`%OS_ROOT%\database\CLOG\CLOG` (Windows)

`${OS_ROOT}/database/CLOG/CLOG` (Solaris)

Creating a Contingency Log

For Open Systems, if the contingency log does not exist, you are asked if you want to create it. If you answer yes, you are asked how big the contingency log should be, that is, how many 4 KB pages it should contain.

In either case, you are first asked how many slots should be defined in the contingency log. The number of slots limits the number of distributed transactions (Fail Safe 1 or 2) that can be in progress or in-doubt at any given time.

If the contingency log already exists the command fails unless the -w argument is specified.

See Also *Managing Backup and Recovery* for your operating environment for more information about TIBCO Object Service Broker backup and recovery.

Arguments

Argument	Description
-d *databasedir*	Override the TIBCO Object Service Broker database directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is extracted from the environment variable *OS_ROOT*. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the default TIBCO Object Service Broker log directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.

Argument	Description
-n *#pages*	Specify the size of the contingency log file, in number of 4 KB pages, that you want to format. If the file does not exist, the format proceeds without displaying a prompt. If it does exist, you must also specify the -w argument to overwrite existing data. This value should be at least 3 times the value specified for *#slots* with the -s argument.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is the file crparm under the TIBCO Object Service Broker database directory %OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris.
-s *#slots*	Specify the number of slots. Must be at most 1/3 the value specified for *#pages* with the -n argument.
-w	Reformat the contingency log, overwriting any data in it.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows and ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Contents
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	The Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	The database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	The Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	The TIBCO Object Service Broker directory.

Related Utilities

- hrntlfps (Format Pagestore)
- hrntlfjr (Format Journal)
- hrntlfrl (Format Redolog)

hrntlfjr (Format Journal)

Syntax hrntlfjr [*arguments*] *journal*

Platforms Windows, Solaris

Description This utility is used to format a journal. The journal must be identified with its full path name, as in the following examples:

```
%OS_ROOT%\database\JOURNAL\JOURNAL1 (Windows)
${OS_ROOT}/database/JOURNAL/JOURNAL1 (Solaris)
```

The actual filename must be of the form JOURNAL*n*, where *n* is 1 or 2.

Creating a Journal

In Open Systems, if the journal does not exist, you are asked if you want to create it. If you answer yes, you are asked how big the journal should be, that is, how many 4 KB pages it should contain.

The journal file must have at least as many pages as there are Resident pages defined in the crparms file. If you try to format a journal file with fewer pages, the utility fails.

If the journal already exists the command fails unless the -w argument is specified.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the default TIBCO Object Service Broker database directory (%OS_ROOT%\database in Windows, ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is the value of the environment variable *OS_ROOT*. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows, ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.

Argument	Description
-n *#pages*	Specify the size of the journal file, in number of 4 KB pages, that you want to format. If the file does not exist, the format proceeds without displaying a prompt. If it does exist, you must also specify the -w argument to overwrite existing data.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument overrides those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows, ${OS_ROOT}/database/crparm in Solaris).
-w	Reformat the journal, overwriting any data in it.

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

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows, ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Contents
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	The Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	The database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	The Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	The TIBCO Object Service Broker directory.

Related Utilities

- hrntlfps (Format Pagestore)
- hrntlfcl (Format Contingency Log)
- hrntlfrl (Format Redolog)

hrntlfps (Format Pagestore)

Syntax hrntlfps [*arguments*] *pagefile1* [*pagefile2* [*pagefile3*...]]

Platforms Windows, Solaris

Description This utility is used to format one or more TDS page data files. The page data files must be given with their full path name, for instance:

```
%OS_ROOT%\database\SEG1\PAGE2 (Windows)
${OS_ROOT}/database/SEG1/PAGE2 (Solaris)
```

The actual filename must be of the form PAGE*n*, where PAGE is written in uppercase and *n* is an integer from 1 to 128.

Creating a Page Data File

In Open Systems, if the page data file does not exist, you are asked if you want to create it. If you answer yes, you are asked how big the page data file should be, that is, how many 4 KB pages it should contain.

If the page data file already exists the command fails unless the -w argument is specified.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the default TIBCO Object Service Broker database directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is the value of the environment variable *OS_ROOT*. If the *OS_ROOT* variable is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.

Argument	Description
-n *#pages*	Specify the size of the Page data files, in number of 4 KB pages, that you want to format. If the file does not exist, the format proceeds without displaying a prompt. If it does exist, you must also specify the -w argument to overwrite existing data.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is the file crparm under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).
-w	Reformat existing or previously formatted page data files, overwriting any data contained within them.

See Also

TIBCO Object Service Broker Parameters for more information on parameters.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Contents
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	The Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	The database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	The Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	The TIBCO Object Service Broker directory.

Related Utilities

- hrntlfcl (Format Contingency Log)
- hrntlfjr (Format Journal)
- hrntlfrl (Format Redolog)

hrntlfrl (Format Redolog)

Syntax hrntlfrl [*arguments*] *redolog*

Platforms Windows, Solaris

Description This command is used to format the redolog. The redolog must be identified with its full path name, for example:

```
%OS_ROOT%\database\REDO\REDOLOG (Windows)
${OS_ROOT}/database/REDO/REDOLOG (Solaris)
```

Creating a Redolog File

In Open Systems, if the redolog file does not exist, you are asked if you want to create it. If you answer yes, you are asked how big the page data file should be, that is, how many 4 KB pages it should contain.

If the redolog already exists the command fails unless the -w argument is specified.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the default TIBCO Object Service Broker database directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is the value of the environment variable *OS_ROOT*. If *OS_ROOT* is not defined, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.
-n #*pages*	Specify the size of the redolog, in number of 4 KB pages, that you want to format. If the file does not exist, the format proceeds without displaying a prompt. If it does exist, you must also specify the -w argument to overwrite existing data.

Argument	Description
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the Data Object Broker parameter file. By default this is under the TIBCO Object Service Broker database directory (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).
-w	Reformat the redolog, overwriting any data in it.

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files

For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Contents
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	The Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	The database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	The Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	The TIBCO Object Service Broker directory.

Related Utilities

- hrntlfps (Format Pagestore)
- hrntlfjr (Format Journal)
- hrntlfcl (Format Contingency Log)

hrntlmrg (Journal Merge)

Syntax hrntlmrg [*arguments*] *spinfile* [*spinfile1* [*spinfile2* [*spinfile3*...]]] *mergefile*

Platforms Windows, Solaris

Description The hrntlmrg utility is used in TIBCO Object Service Broker backup and recovery procedures. This command is used to merge journal spin files with a segment backup file to create a more up-to-date backup file. In general, this command merges any files containing pages from TDS page data files (that is, any files generated by the hrntlbps (Backup Pagestore) utility or by the hrnspjex (Journal Extraction (or Journal Spin)) utility. Only the most recent image of each page is kept; all other pages are discarded.

The backup file can contain pages from one or more segments as can all the spin files. The -s argument can be used to filter pages based on segment number.

spinfile and *mergefile* must not be the same filename. The program fails if the two names are identical.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the default TIBCO Object Service Broker database directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is the value of the environment variable *OS_ROOT*. If *OS_ROOT* is not defined, this argument must be specified.
-l *logdir*	Override the default TIBCO Object Service Broker log directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`

Argument	Description
-p *parmfile*	Override the default Data Object Broker parameter file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).
-s *segnumber*	Include only those pages associated with segment *segnumber* in the output file. Otherwise all pages are merged into the output file.
-w	Overwrite *mergefile* if it exists.

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

Environment Variables

OS_ROOT

Set this environment variable to indicate the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utilities

- hrntlbps (Backup Pagestore)
- hrntlrps (Restore Pagestore)
- hrnspjex (Journal Extraction (or Journal Spin))

hrntlpcl (Print Contingency Log)

Syntax hrntlpcl *clogfile*

Platforms Windows, Solaris

Description hrntlpcl prints the contingency log.

Before you print the contingency log, you must shut down the Data Object Broker. The output from this utility gives you detailed information about each active transaction on the contingency log at the time of shutdown.

Arguments

Argument	Description
clogfile	The full path and name of the contingency log file you want to print.

Sample Output

```
...

Contingency Log Print

Track 0004  Wed Mar 27 07:01:26 2002

    Stage CONTINGT  Status IN-DOUBT  commit# 835  UserId USR40
    Local TrxId 01000B7E REMOTE:PEER

    0000  002A0202 0001D900  000B7E00 01001BD9    |......R...=....R|
    0010  0CE3C3F0 F0F3F1F3  F2E3C1F0 F1000D01    |.TC003132TA01...|
    0020  0204D7C1 E4D30307  000C                 |..PAUL..........|

    Peer Trx Id 00000B7E  Lock Type D9  Number of Updates 1

    REPLACE for table TC003132TA01

    0000  000D0102 04D7C1E4  D3030700 0C          |.....PAUL.......|
```

hrntlrps (Restore Pagestore)

Syntax hrntlrps [*arguments*] *backupfile*

Platforms Windows, Solaris

Description You can use the hrntlrps utility as part of the TIBCO Object Service Broker recovery process. This utility restores a TDS segment from a backup file created by the hrntlbps (Backup Pagestore) utility. The segment must be offline. Any data currently in the segment is overwritten.

It can also be used to apply to a partial backup on a segment, as part of operational maintenance. This is done using an optional -P parameter.

If the -s argument is not specified, the hrntlrps utility restores segment 0 (MetaStor) by default.

You can specify whether to restore (-r) and/or verify (-v) a file. If neither of these arguments are given, the file is both validated and restored by default.

See Also *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery.*

Arguments

Argument	Description
-d *databasedir*	Override the default TIBCO Object Service Broker database directory (%OS_ROOT%\database in Windows or ${OS_ROOT}/database in Solaris).
-h *hurondir*	Override the TIBCO Object Service Broker installation directory. By default this is the value of the environment variable *OS_ROOT*. If *OS_ROOT* is not set, this argument must be specified.
-l *logdir*	Override the TIBCO Object Service Broker log directory. By default this is under the TIBCO Object Service Broker installation directory (%OS_ROOT%\log in Windows or ${OS_ROOT}/log in Solaris). This argument is ignored if the Data Object Broker is active.

Argument	Description
-o *parmoverrides*	Override one or more Data Object Broker parameters. Data Object Broker parameters are normally set in the crparm file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris). Parameters set with this argument override those in the crparm file. The syntax is: `parm1=val1[,parm2=val2[,parm3=val3[...]]]`
-p *parmfile*	Override the default Data Object Broker parameter file (%OS_ROOT%\database\crparm in Windows or ${OS_ROOT}/database/crparm in Solaris).
-P	Indicates that the partial backup can be accepted.
-r	Restore the *backupfile*.
-s *segnumber* or *segname*	Restore segment *segnumber* or *segname* from *backupfile*.
-v	Validate the segment data for the specified segment in the *backupfile*.
backupfile	The name of the backup file from which the TDS segment is to be restored.

Use the `-P` parameter with caution. If not used properly, it can corrupt your MetaStor.

Environment Variables

OS_ROOT

Set to the directory where TIBCO Object Service Broker is installed.

HURONDIR

Set this environment variable if the TIBCO Object Service Broker directory is not located in the default location (%OS_ROOT%\database\huron.dir in Windows or ${OS_ROOT}/database/huron.dir in Solaris).

Files For a full description of these files, refer to Appendix A, TIBCO Object Service Broker Files, on page 101.

Windows	Solaris	Description
%OS_ROOT%\database\crparm	${OS_ROOT}/database/crparm	Holds the Data Object Broker parameter files.
%OS_ROOT%\database\dbdef	${OS_ROOT}/database/dbdef	Holds the database definition.
%OS_ROOT%\log\hrncr.*	${OS_ROOT}/log/hrncr.*	Holds the Data Object Broker log file.
%OS_ROOT%\database\huron.dir	${OS_ROOT}/database/huron.dir	Holds the TIBCO Object Service Broker directory.

Related Utilities

- hrntlbps (Backup Pagestore)
- hrnspjex (Journal Extraction (or Journal Spin))
- hrntlmrg (Journal Merge)
- hrnbrext (Extract Selected Pages)

htrans (Translates z/OS Files To and From TIBCO Object Service Broker Binary Files)

Syntax htrans [-a] [-f *output*] *input*

Platforms Windows, Solaris

Description The htrans utility translates ASCII files downloaded from a z/OS system for use with the LOAD tool in TIBCO Object Service Broker. It can also translate binary files generated by the UNLOAD tool in TIBCO Object Service Broker for uploading to a z/OS system.

Arguments

Argument	Description
-a	Translate a formatted TIBCO Object Service Broker file into ASCII for uploading to the a z/OS system.
-f *output*	Specifies the name of the output file.
input	Specifies the name of the input file.

Be sure to specify the input and output files in the correct order. Otherwise, htrans overwrites your input file.

Examples These are some examples using the htrans utility:

- `htrans transfer.db`

  Translates the file transfer.db, which was downloaded from a z/OS system, into TIBCO Object Service Broker database files in %OS_ROOT%\database (Windows) or ${OS_ROOT}/database (Solaris).
- `htrans -f IMPORT.FIL transfer.imp`

  Translates the file transfer.imp (which was downloaded from a z/OS system) to a TIBCO Object Service Broker type file IMPORT.FIL.
- `htrans -a -f upload.fil OUT.DAT`

  Translates OUT.DAT, a TIBCO Object Service Broker type file generated from TIBCO Object Service Broker, to an ASCII file suitable for uploading to a z/OS system.

osBatch (Start a TIBCO Object Service Broker Batch Client)

Syntax osBatch {*parameter*[=*value*] [, *parameter*[=*value*]]}

Platforms Windows, Solaris

Description Executing osBatch starts a single-session Execution Environment. The rule specified by the RULE parameter is executed. When this rule is finished, the session ends and the Execution Environment exits. The exit code of the osBatch process is 0 unless the $SETSESSIONEND tool is used to set the session exit code, or the session abends. If the session abends, the exit code is greater than 127.

To determine the cause of an abend, refer to *TIBCO Object Service Broker Messages With Identifiers*.

An osBatch session normally creates an Execution Environment log file and a session log file, in install_path/log for Solaris and in install_path\log for Windows. The default name of the Execution Environment log file is EE.<configuration name>.<date>-<time>.log. The default name of the session log file is batch.<configuration name>.<date>-<time>. log. The value for <configuration name> is determined by the EENAME parameter for the Execution Environment log file, and the NAME parameter for the session log file.

While osBatch is setting up, it writes parameter syntax and validation error messages to the console and to *startup.<date>.log* (for example, startup.04.24.log created on April 24). For example, the command line `osBatch aaa=bbb` results in some console output, which also appears also in the log file, reporting that the parameter is invalid.

Session and Execution Environment parameters are used to control the behavior of a batch session. Refer to *TIBCO Object Service Broker Parameters* for detailed information about parameters. The values can be assigned to the osBatch parameters from the following sources (in order of precedence):

Source	Description	Precedence
Command line	Both Execution Environment and session parameters can be specified.	Highest
User options	Session parameters specified on the user's profile screen.	
Parameter file	Execution Environment parameters specified on the user's profile screen.	

Source	Description	Precedence
Registry (Windows only)	Execution Environment and session parameters read from the Windows registry.	
Environment variables	Execution Environment and session parameters read from environment variables.	
Default values	Execution Environment and session parameters have reasonable default values. **Note** *In most cases, you must override the parameters shown in the following table.*	Lowest

Execution Environment Parameters Requiring Override	Description
DOB	Name of the Data Object Broker to which the Execution Environment connects.
USERID	user ID.
PASSWORD	User password.
RULE	Name of rule to be executed.
SEARCH	SEARCH defaults to S for SITE. If your rule is located in your local library, set SEARCH to L.
LIBRARY	Library that contains the rule to execute.
BATCHCONSOLE/ NOBATCHCONSOLE	BATCHCONSOLE: osBatch output (except for the copyright message followed by the parameter list) appears on the console as well as in the log files. NOBATCHCONSOLE: no output appears on the console, except for parameter syntax errors. Default: NOBATCHCONSOLE.
BROWSE/ NOBROWSE	If your rule updates database tables, set this to NOBROWSE. Default: NOBROWSE.

Parameters To access parameter settings of parameter files, use the registry or environment variables NAME or EENAME.

To access session parameter settings, use the NAME parameter.

To access Execution Environment parameter settings, use the EENAME parameter.

Refer to *TIBCO Object Service Broker Parameters* for more information on parameters.

Example 1- Command Line

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

Since the command line has the highest precedence, the values of the specified parameters are used during the execution of the batch session. Other Execution Environment and session parameters can be set in their respective DEFAULT configurations.

Example 2 - Session.prm file

```
osBatch NAME=BATCH2
```

Suppose the session.prm file contains the following:

```
NAME=BATCH2
EENAME=BATCHEE
USER=JOEB
P=XYZ123
R=BATCH1
SEARCH=L
NOBROWSE
LIBRARY=PRODBATCH
```

... and the ee.prm file contains the following:

```
NAME=BATCHEE
DOB=PROD
```

This execution of osBatch assigns the same values to the DOB, USER, PASSWORD, RULE, SEARCH, BROWSE and LIBRARY parameters as in Example 1.

In Example 2, the source of the parameter assignments is the parameter file instead of the command line.

Example 3 - Command Line

```
osBatch NAME=BATCH2 DOB=TEST
```

If the ee.prm and session.prm are set as in Example 2, all the parameters are set to the same values they are assigned in Example 2 with the exception of the Data Object Broker parameter. Since the command line has the highest precedence, it overrides the value assigned from the ee.prm parameter file.

osMonSvc (Install osMon as a Windows Service)

Syntax osMonSvc [*arguments*]

Platform Windows

Description This command is used to install the TIBCO Object Service Broker monitor (osMon) as a Windows service, as well as update the appropriate Windows Registry entries. After installation, the osMon service has a startup type of "Manual". You can change this option to "Automatic" so that the service starts each time you start Windows.

See Also *TIBCO Object Service Broker for Open Systems Installing and Operating* for more information on operating osMon as a Windows service.

Arguments

Argument	Description
-h *huronpathdir*	Overrides the TIBCO Object Service Broker installation directory. By default this is extracted from the environment variable *OS_ROOT*. If the *OS_ROOT* variable is not set, this argument must be specified.
-i *svcname*	Installs osMon as a Windows service, with the name specified, and creates the required Windows Registry entries. The default for the name is the name of the utility as you type it in to run it.
-o *parmoverrides*	Overrides one or more Execution Environment parameters of type "TIBCO Object Service Broker monitor". These parameters are normally set in the mon.prm file (%OS_ROOT%\database\mon.prm). The syntax is: `parm1=val[,parm2=val[,parm3=val[...]]]`
-u *svcname*	Uninstalls the osMon service specified and removes the associated Windows Registry entries.
-v *svcname*	Displays whether the osMon service specified is installed.

rsview (Data Object Broker Resource Viewer)

Syntax rsview

Platforms Windows, Solaris

Description Using rsview, an administrator can display on standard out (stdout) the resources and servers for a Data Object Broker.

Usage Notes

- The rsview utility has no arguments and must be run on the same machine as the Data Object Broker.
- The *OS_ROOT* and *HURONDIR* environment variables determine the Data Object Broker under review.

Sample Output

```
...
Data Object Broker Resource Viewer
Executed: Wed Mar 22 15:33:48 2007
Resources for TESTA
RMT-DOB         CONID PEERID    STATUS      USERID   TXNO    DATE       TIME
DEVTEST         00000  WB00001 CONNECTED   SYSADMIN 00000007 2007/03/20 14:57:54
DEVTEST 00001      00001  WB00002 CONNECTED
Servers for TESTA
TYP SRVR-ID SRVR-USR RMT-DOB         RMT-USR SRV-TXNO RMT-TXNO DATE       TIME
PRS DEFAULT0 TDOZE401 DEVTEST        HURON   0000000b 0000000a 2007/03/20 14:57:54
PRS DEFAULT0 TDOZE402
End of resources and servers list for TESTA
```

Description of the Report Output

As illustrated, the Resource portion of the output displays:

- The remote Data Object Broker name, for example, DEVTEST.
- The connection ID that appears in error messages in the Data Object Broker log, for example, 00000.
- The peer user, for example WB00001, who is currently logged in to the remote Data Object Broker DEBTEST from TESTA.
- The status of the connection can be CONNECTED, DISCONNECTED, or UNKNOWN.
- The local user that is currently using the resource, for example, SYSADMIN. This is left blank if the connection is not being used.
- The identifier of the transaction that is using the resource, for example, 00000007. This is left blank if the connection is not being used.

- The date and time of the transmission of the last message from the current transaction.

The Server portion of the output displays the state of the inbound PEER and SERVER connections for the Data Object Broker, and includes:

- The type of the server, for example, PRS for a PEER server.
- The identifier of the server that appears in the Data Object Broker log, for example, DEFAULT0.
- The user ID the server uses to access the Data Object Broker, for example, TD0ZE401.
- The name of the remote Data Object Broker, for example, DEVTEST. This is left blank if the connection is not being used.
- The name of the remote user, for example, HURON. This is left blank if the connection is not being used.
- The local transaction ID, for example, 000000b. This is left blank if the connection is not being used.
- The transaction ID for the remote user, for example, 000000a. This is left blank if the connection is not being used.
- The date and time of the transmission of the last message from the current transaction.

Appendix A TIBCO Object Service Broker Files

This appendix describes the files used by TIBCO Object Service Broker.

Topics

- Description of the Files, page 102

Description of the Files

Data Object Broker Parameter Files

File Name Either one of the following:

- %OS_ROOT%\database\crparm (Windows)
- ${OS_ROOT}/database/crparm (Solaris)

Description This file allows the TIBCO Object Service Broker administrator to tailor the operation of the Data Object Broker. Some of the parameters are used to handle operating system differences, others are used to set the expected work load and throughput, while still others control such functions as logging and tracing. Refer to *TIBCO Object Service Broker Parameters* for more information about Parameters.

Form The parameters are all of the form:

```
PARAMETER=value
```

Usage Notes

- Before changing the value of a parameter in this file, you must shut down the Data Object Broker.
- Only one parameter is allowed on each line.
- Blank lines and white space are ignored.
- Comments begin with the number sign (#) character and continue to the end of the line.
- The parameter names and values are not case sensitive.

Database Definition

File Name Either one of the following:

- %OS_ROOT%\database\dbdef (Windows)
- ${OS_ROOT}/database/dbdef (Solaris)

Description This file is used to define the segments, journals, and logs that comprise the TIBCO Object Service Broker database. Primarily this means defining the name of each segment (and thereby the subdirectory) and the number of files in which the segment is stored.

Form

The dbdef file consists of two types of lines:

DB or main line	Defines properties and directory paths for default file locations. Each DB line contains one definition. Blank lines and white space are ignored. Comments begin with the # character and continue to the end of the line. The keyword names and values (except for the PATH and FILE values on the Solaris platform) are case insensitive. Each Data Object Broker special file is described by a single DB (or main) line in the dbdef file. Among other properties, the DB line can define the default path for all files that belong to the special file. The DB line uses the following syntax: `DB TYPE=type,NAME=name[,ACBS=numfiles] [,ID=segno] [,INIT=initflag] [,PATH=pathstring]`
FILE subcommand linde	Specifies the directory path to a file described in the preceding DB line, overriding the default path to the file. FILE lines are optional. The filename is determined by the NAME parameter in the DB line. The FILE line uses the following syntax: `FILE FILENO=n [,PATH=pathstring]`

Variable Descriptions

type	CLOG, JOURNAL, PAGE, or REDOLOG.
name	The name of the Data Object Broker special file. This is the name of the subdirectory (under database) that contains the segment, journal, or log files. The name is converted to uppercase.

numfiles	The number of separate files involved, depending on the value of TYPE and NAME, as shown here:	
	TYPE=CLOG	ACBS=1
	TYPE=JOURNAL	ACBS=2
	TYPE=PAGE, NAME=METASTOR	ACBS=3
	TYPE=PAGE, NAME=AUDITLOG	ACBS=1
	TYPE=PAGE, NAME=PageFileName	ACBS=the number of page files associated with the segment, to a maximum of 128. This number depends on the number of disks available for the database (Solaris) or the actual number of page files required (Windows).
	TYPE=REDOLOG	ACBS=1
segno	The segment number, when TYPE=PAGE. The range of valid values is 0 (for the MetaStor) to 255.	
initflag	When TYPE=PAGE, this determines if the segment should be online after Data Object Broker startup. Valid values are YES and NO. The alternative is to have the TIBCO Object Service Broker system administrator activate the segment explicitly using the dbonline operator command. The INIT value for a segment can be changed at any time, for the next restart of the Data Object Broker, except for segment 0, which must be available at all times.	
path_string	The default (on the DB line) or the actual (on the FILE line) directory path for the segment or log.	
n	Range from 1 to 128. This value cannot be greater than the ACBS value on the corresponding DB line.	

Usage Notes

- Only one definition is allowed on each line.
- Blank lines and white space are ignored.
- Comments begin with the hash (#) character and continue to the end of the line.

- The keyword names and values (except for the PATH and FILE values on the Solaris platform) are case insensitive.
- To increase the number of page files in your Pagestore, increase the ACBS value accordingly *before* formatting the new page files.

Example

```
DB     TYPE=PAGE,ACBS=4,NAME=METASTOR,INIT=YES
FILE   FILENO=1,PATH=D:\MetaStor1\
FILE   FILENO=2,PATH=D:\MetaStor2\
FILE   FILENO=3,PATH=D:\MetaStor3\
FILE   FILENO=4,PATH=D:\MetaStor4\
```

Data Object Broker Log File

File Name

Either one of the following:

- %OS_ROOT%\log\hrncr.* (Windows)
- ${OS_ROOT}/log/hrncr.* (Solaris)

Description

This file is used to log Data Object Broker activity and problems. It is normally one of the first places to look if a problem involving the Data Object Broker is suspected.

Usage Notes

- Every time the Data Object Broker is started it creates a log file in the log directory.
- The name of this file is platform-dependent.

Windows	hrncr.*nnn*	*nnn* is an incrementally sequential counter that uniquely identifies the log file.
Solaris	hrncr.*nnnnn*	*nnnnn* is the process ID of the Data Object Broker supervisor—the background hrncr process.

TIBCO Object Service Broker Directory

File name Either one of the following:

- %OS_ROOT%\database\huron.dir (Windows)
- ${OS_ROOT}/database/huron.dir (Solaris)

Description This file is used to define a list of TIBCO Object Service Broker nodes. The directory is used by TIBCO Object Service Broker Communication Support (OCS). By default the directory is located in *%OS_ROOT%*\database\huron.dir, but this location can be overridden by the *HURONDIR* environment variable. Refer to *TIBCO Object Service Broker Parameters* for more information about Parameters.

Form Each entry has the following form:

```
NODE NAME=longname,
ALIAS=shortname,
HOST=hostname,
IP=ipaddress,
SERVICE=servicename,
PORT=portnumber,
IPCKEY=ipckey,
TIMEOUT=timeoutvalue
```

Variable Descriptions

longname	The full name of the node. 1 to 16 alphanumeric characters. Corresponds to the Data Object Broker NODENAME parameter. If *longname* is longer than eight characters, an ALIAS must also be specified.
shortname	A shortened version of the node name. 1 to 8 alphanumeric characters. Required if the node name is longer than eight characters.
hostname	The name of the TCP/IP host where the node resides. Use the fully qualified Internet host name, or the alias used within TCP/IP. Mandatory unless the node resides on z/OS.
ipaddress	The TCP/IP address of the host. Specified using decimal delimiters (for example, 126.0.0.56). This should be used only if the host name is not resolvable.
servicename	The service name used by this node for TCP/IP communication. 1 to 8 alphanumeric characters.
portnumber	The TCP/IP port where this Data Object Broker listens for incoming connections (should be 10000 or higher).

ipckey	Identifier used for semaphores and shared memory communications. Must not conflict with other Data Object Brokers running on the same system.
timeoutvalue	Optional timeout value specifying the timeout time for unresponsive outbound connections to a server. The range is 0 to 2,147,483,647 milliseconds. This parameter is implemented only for Windows; it is ignored in a Solaris environment. **Note** Only for the *hrncr (TIBCO Object Service Broker Data Object Broker)* utility and the *hrntladm (Administration Menu)* utility

Usage Notes

- All keywords and values can be specified in upper or lowercase—the data is not case sensitive.
- An entry can be contained on one line or spread over several by ensuring that the last character on a line that is not whitespace is a comma.
- Comments begin with the hash (#) character and continue to the end of the line.

Appendix B Defining Batch Load Control Cards Manually

This appendix describes how to define batch load control cards manually.

Topics

- Overview, page 110
- Specification Cards, page 111
- Page Fill Tailoring, page 113
- Input Definition Cards, page 114
- Output Definition Cards, page 116
- Value Cards, page 118

Overview

Manually Defining Control Cards

This chapter explains how you can define your control cards manually. In most cases you can use the automated control card preparation facility as described in the documentation for BATCHLOAD_CARDS in *TIBCO Object Service Broker Shareable Tools*. After building the control card file, you can use the information in this chapter to locate and change any particular field if necessary.

Types of Control Cards

The hrnbrtbl (Batch Load) utility requires up to four types of control cards:

- Specification cards
- Input definition cards
- Output definition cards
- Value cards

The control card types must be stored in the file in this order.

Keyboard Constraint

It is possible that manual definition of control cards does not work when using a non-U.S. keyboard. Use of other keyboards can result in transcription of the table name and other identifiers in another code base.

Specification Cards

Specification cards are used to specify override values for the batch load process. Unless it is omitted, the specification card must be the first card in the control card file.

Field Values

Card Type (column 1)

The card type for a specification card is S. If the specification card is omitted, default values are used.

Data Page Fill Level (columns 3 – 4)

The percentage of data space on a data page that is used. If omitted, up to 75 percent of the data space within the page is used. For more information about adjusting these levels, refer to Page Fill Tailoring on page 113.

Primary Index Page Fill Level (columns 6 – 7)

The percentage of data space on a primary index page that is used. If omitted, the value defaults to 75 percent. For more information about page fill levels, refer to Page Fill Tailoring on page 113.

Secondary Index Page Fill Level (columns 9 – 10)

The percentage of data space on a secondary index page that is used. If omitted, the value defaults to 75 percent. For more information about page fill levels, refer to Page Fill Tailoring on page 113.

Group Index Page Fill Level (columns 12 – 13)

The percentage of data space on a group index page that is used. If omitted, the value defaults to 75 percent. For more information about page fill levels, refer to Page Fill Tailoring on page 113.

Total Number of Fields (columns 31 – 33)

The value for this field is the total of:

- The number of input fields

- The number of output parameters and fields that are not derived from the input file

  If omitted, the value defaults to 50. The value is used when calculating storage requirements for the internal definition tables.

Dynamic Unit Name (columns 35 – 42)

Should always be SYSDA.

Page Fill Tailoring

Tailoring Guidelines

With the hrnbrtbl (Batch Load) utility you can manipulate the percentage of usable page space that is filled during the load process. By adjusting the page fill values, you can create page structures that are better suited to their intended purposes. For example, if the table you want to load is very stable and you expect few or no updates to be applied (for example, a history instance of sales records), you could increase the fill percentage. However, if you expect your table to have much occurrence expansion, you could decrease the fill percentage to minimize the number of page splits.

Page Split/Merge Processing

During normal online processing, if a page is too full to accept an update or insert, the page is split dynamically. If two consecutive pages can hold all the data on one page after a delete, the pages could be merged dynamically. This dynamic split/merge process alleviates the need for offline page restructuring, but can slow down your processing. If you adjust the fill percentages to meet your specific intentions, page split/merge processing is reduced.

Input Definition Cards

The input definition card describes the layout of the user input record. Each field on the input record must be defined, whether or not it is to be loaded into the TIBCO Object Service Broker table.

Field Values

CARD TYPE (column 1)

The card type for the input definition card is I.

SEQUENCE # (columns 3 – 5)

The cards for the input definition must arrive in the order the fields appear on the input record. The sequence numbers are defined to ensure this sequence is maintained.

ENTRY TYPE (column 7)

There are two possible values for the entry type of input cards:

R (Record)	The card specifies the file as variable or fixed length. An indicator of V or F is set in the type field (column 46). Entry type R cards must come before entry type F cards.
F (Field)	A field definition is defined on the card.

If the record type card is omitted, the input file is assumed to be fixed length.

NAME (columns 9 – 44) (Input Definition ENTRY TYPE F only)

The name field is defined to provide a cross reference between user input and TIBCO Object Service Broker output field names. The field is 36 bytes long.

TYPE (column 46) (Input Definition ENTRY TYPE R only)

The file type. Must be V (variable-length) or F (fixed-length).

SYNTAX (columns 45 – 48) (Input Definition ENTRY TYPE F only)

The syntax field is used to indicate the syntax of the input field. The syntax is left-justified if it is a raw-data or Unicode syntax; otherwise, it is right-justified.

If this syntax is either B (binary) or P (packed), you must enter the null equivalent value in the cross-referenced target field, **Output Definition Entry Type F entry**. If left blank, the default null equivalent value takes effect. For more information, refer to Appendix C, Null Handling, on page 119.

If you want to load data directly into a field with a semantic type of date, you must define your input data field with a syntax of C (alphanumeric) and the data must be in your site default format (for example, 1998-03-01).

Field Length (columns 49 – 52) (Input Definition ENTRY TYPE F only)

This field specifies the length of an input field. Refer to *TIBCO Object Service Broker Managing Data* for information on permitted lengths.

Decimal Position (columns 54 – 55) (Input Definition ENTRY TYPE F only)

This field defines the number of post decimal positions for a numeric field. If omitted, the decimal position defaults to 0.

Decimal positions are ignored for character fields.

Offset (columns 59 – 62) (Input Definition ENTRY TYPE F only)

The offset defines the location of the field within the input record. The field is optional. If omitted, the offset is calculated.

Table Name (columns 64 – 79) (Input Definition ENTRY TYPE F only)

The specification of **Table Name** is required to allow the program to match input and output fields. This field should contain the field name in the TIBCO Object Service Broker TDS table definition that cross-references the input field name. If omitted, the input field is ignored.

Output Definition Cards

The output definition card describes the layout of the TIBCO Object Service Broker table.

Field Values

Card Type (column 1)

The Card Type for the TIBCO Object Service Broker output definition is H.

Sequence # (columns 3 – 5)

The cards for the output definition must define fields in the order they appear in the TIBCO Object Service Broker table. The sequence number is defined to ensure this sequence is maintained. The output definitions for the parameters (for TDS only) must arrive before the output definitions for the fields.

Entry Type (column 7)

There are three possible values for entry type:

R (Record)	The R type entry is used to define the character set (columns 9-12), and the TIBCO Object Service Broker table name using the TIBCO Object Service Broker name field (columns 64-79). For a list of valid character set identifiers, refer to the documentation for BATCHLOAD_CARDS in *TIBCO Object Service Broker Shareable Tools*. Entry type R cards must come before P or F cards.
P (Parameter)	If the entry type is P, a parameter is defined.
F (Field)	If the entry type is F, a field is defined.

Name (columns 9 – 24)

The name of the TIBCO Object Service Broker parameter (for TDS only) or field.

Type or Syntax (columns 45 – 46)

These columns contain the syntax if the field is raw data or Unicode; otherwise, the semantic type of the parameter or field goes in column 46.

Syntax (column 48)

The syntax of the output parameter or field, for fields that are neither RD nor UN.

Maximum Length (columns 49 – 52)

The maximum length of the output parameter or field.

Decimal Position (columns 54 – 55)

The decimal position specified (that is, the number of post decimal positions) for a numeric parameter or field. If omitted, the decimal position defaults to 0.

Decimal positions can be specified only for packed (syntax P) fields.

Key Type (column 57)

This field specifies the TIBCO Object Service Broker field key type indicator for a given field. The key type indicator is one of the following values:

- P – Primary
- S – Secondary
- Q – Both primary and secondary
- blank – Non-key

- Primary key fields must be either cross-referenced to an input field or system generated (that is, an IDgen key). They must not appear on value cards.
- If a primary key field is defined as binary length 4 and is system generated (that is, an IDgen key), it must be the only key.

Null-Equivalent Value (columns 64 – 79)

This field applies only to input fields defined with either syntax B or P. Refer to "Input Definition Cards" for more information.

Refer to Appendix C, Null Handling, on page 119 for more information.

Value Cards

If you require a field in TIBCO Object Service Broker that does not exist in the user input file, you can specify a value card to assign a default value to the field. The value card provides a means of setting a default value for any parameter or non-primary-key field that is present in TIBCO Object Service Broker but is not derived from the input file.

Value cards cannot be used for fields with syntax RD or UN.

Field Values

Card Type (column 1)

The Card Type of a value card is V.

Name (columns 3 – 18)

The name of the TIBCO Object Service Broker parameter or field to which the value is assigned.

Continuation Number (column 20)

Each value card provides up to 50 bytes of text value. You can use the continuation number to continue the text for up to 6 cards.

The sequence of the continuation is verified to ensure the cards are ordered properly. The sequence is 0, 1, 2, 3, 4, 5. You can leave the continuation number blank instead of using a zero. All 50 bytes of the text of the card are used up to the maximum length of the field.

Text Value (columns 22 – 71)

The text area contains the value to be used. If the value is numeric (binary or packed TIBCO Object Service Broker parameter or field), the text is entered in character format. If the value is negative, the first byte of the value (column 22) must be a minus sign (that is, -123).

Appendix C Null Handling

This appendix describes how to handle null values.

Topics

- Syntax Specific Nulls, page 120
- Alternative Null Equivalents, page 122
- -O Option (Batch Unloads hrnbrulb and hrnbrulh Only), page 123

Syntax Specific Nulls

Utilities moving data from a TIBCO Object Service Broker table to an external file or from a file to a TIBCO Object Service Broker table must be able to preserve null values. This section describes the strategies used to store null values in syntax-specific files that are external to TIBCO Object Service Broker.

Syntax Types V, F, and C

Fields containing binary zeroes are considered to be nulls for syntax types F, RD, UN, and V. For syntax C, a field containing all spaces is considered to be null.

Syntax Types B and P Only

By default, if a type B or P field contains the lowest possible value that it can hold, the field is considered to have a null value.

Binary Fields

For binary fields (syntax type B) the following values are considered to be equivalent to nulls:

4-byte B-type field	-2147483648 (X'80000000')
2-byte B-type field	-32768 (X'8000')

Packed Decimal Fields

For packed decimal fields, although the lowest possible value is used as the null-equivalent, the alternate negative sign `X'B'` is used. This means that the lowest possible value can still be used in tables (TIBCO Object Service Broker uses only the standard negative sign `X'D'`).

In a file where the default null-equivalent is to be used in the load process, the following values must not be used in a packed field except to deliberately indicate nulls.

8-byte P-type field	X'999999999999999B'
7-byte P-type field	X'9999999999999B'
6-byte P-type field	X'99999999999B'

5-byte P-type field	X’999999999B’
4-byte P-type field	X’9999999B’
3-byte P-type field	X’99999B’
2-byte P-type field	X’999B’
1-byte P-type field	X’9B’

Semantic Type D (Date) Fields

The value X’80000000’ for a four-byte date is not valid, so it never occurs in a date field. It is therefore a safe null-equivalent value.

The value shown for a two-byte date represents 1890-04-14 (YYYY-MM-DD); therefore, this date is not usable if the default null-equivalent is used.

Alternative Null Equivalents

Alternatives to the default null-handling behavior are applicable only for syntax B and P. Possible alternatives include:

- Highest possible value as the null equivalent—HIGHVALUE

4-byte B-type field	2147483647 (X'7FF...');
2-byte B-type field	32767 (X'7FFF')
P-type field	X'99.....9A'

  For packed decimal the alternate positive sign (X'A') is used.

  Enter `HIGHVALUE` in columns 64-74.

- User-defined value as the null equivalent

  For packed decimal the standard signs—X'C' for positive and X'D' are used.

  You must enter the null equivalent that you selected in the Null-Equivalent field (columns 64-79) of the appropriate control record.

- No Null Equivalence permitted—NOVALUE

  Enter `NOVALUE` in columns 64-79.

When unloading TIBCO Object Service Broker tables, if the null equivalent value is discovered in the table, it is considered to be an error. Similarly, if a null is discovered in a **NOVALUE** field in the table being unloaded, this error causes the unload utility to abort.

-O Option (Batch Unloads hrnbrulb and hrnbrulh Only)

If this parameter is specified, the unload utilities unload null values as zeroes. This parameter allows for compatibility with earlier TIBCO Object Service Broker releases and for tables being unloaded but not intended for reloading.

Example

The following illustrates the usage of the -O option:

```
hrnbrulb -A auditfile -C controlfile -s Segment# -O unloadfile
```

Floating Point Nulls

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using hrnbrulb or hrnbrulh, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using hrnbrtbl, they are loaded as a zero.

Index

Symbols

?, hrncr subcommand 62

A

ACBS parameter, dbdef file 104
ACCESSLOG, moving off segment 0 11
Administration Menu utility
  arguments 72
  description 72
  environment variables 73
audit log, sample
  hrnbrbal 7
  hrnbrial 13
  hrnbrpgc 19
  hrnbrset 29
  hrnbrsix 32
  hrnbrtbl 39
  hrnbrula 46
  hrnbrulb 52
  hrnbrulh 60
audit report, sample
  hrnbrptr 24

B

Backup Pagestore utility
  arguments 74
  description 74
  environment variables 75
base segment, using hrnbrclr to clear table in 9
Batch Load utility
  arguments 33
  constraints 35
  control file 35
  description 33
  loading TDS table with 33
  sample audit log 39
Batch Pointer Check utility
  arguments 21
  description 20
  error log file 22
  return codes 6, 24
  sample audit report 24
Batch Secondary Index Build utility
  arguments 30
  constraints 30
  control file 30
  description 30
  sample audit log 32
Batch Segment Re-initialization utility
  arguments 26
  constraints 26
  description 26
  environment variables 27
  prerequisites 26
  sample audit log 29
Batch Table Clear utility
  arguments 9
  constraints 9
  description 9
Batch Unload (offline) utility
  arguments 48
  constraints 49
  control file 49
  description 48
  sample audit log 52

Batch Unload (online) utility
  arguments 55
  constraints 56
  control file 56
  description 54
  sample audit log 60
Batch Unload from Archive utility
  arguments 41
  constraints 42
  control file 42
  description 40
  sample audit log 46
Batch Unload utility
  -O option 123
BATCHLOAD_CARDS tool
  and hrnbrtbl utility 35
BATCHUNLD_CARDS tool
  and hrnbrulb utility 49
  and hrnbrulh utility 56

C

canceluser, hrncr subcommand 62
checkpoint, hrncr subcommand 62
clearing
  offline table with hrnbrclr utility 9
  online table with $CLRTAB tool 9
  table in MetaStor segment using hrnbrclr utility 9
  transient data from a TDS segment 26
contingency log
  printing 89
continuous backup and hrnbrset 27
control cards
  defining manually 109
crparm 102
customer support xvi

D

Data Object Broker log file 105
Data Object Broker parameter file 102
Data Object Broker utility
  arguments 61, 66
  command to operate 61
  description 61
  environment variables 64
  subcommands 62
data, loading into predefined tables 33
database definition file 102
DB line, dbdef file 103
dbdef 102
  as input to hrnbrptr 20
dboffline, hrncr subcommand 62
dbonline, hrncr subcommand 62
defining
  input definition cards 114
  output definition cards 116
  specification cards 111
  value cards 118
deleting selected table instance using hrnbrclr utility 9

E

Extraxt Selected Pages utility 10
  environmental variables 10

F

FILE subcommand line, dbdef file 103
FILENO parameter, dbdef file 104
files
  TIBCO Object Service Broker files, description
    Data Object Broker Parameter, description 102
Format Contingency Log utility
  arguments 76
  description 76
  environment variables 77
Format Journal utility
  arguments 79
  description 79
  environment variables 80

Format Pagestore utility
arguments 82
description 82
environment variables 83
Format Redolog utility
arguments 85
description 85
environment variables 86

G

guidelines for page fill tailoring 113

H

hrnbrbal utility
arguments 4
return codes 6
sample audit log 7
hrnbrclr utility
arguments 9
compared with CLRTAB tool 9
constraints 9
description 9
segment requirements 9
selected table instance, deleting 9
hrnbrial utility
arguments 12
description 11
prerequisites 11
sample audit log 13
hrnbrnls utility
arguments 14
constraints 15
control file 15
description 14
return codes 16
hrnbrpgc utility
arguments 18
description 18
sample audit log 19
hrnbrptr utility
arguments 21
dbdef usage 20
description 20
error log file 22
input to 20, 20
orphan pages, defined 22
return codes 24
sample audit report 24
segment requirements 20
hrnbrset utility
arguments 26
constraints 26
description 26
environment variables 27
prerequisites 26
sample audit log 29
hrnbrsix utility
arguments 30
building control file with SIXBUILD_CARDS tool 30
building multiple secondary indexes 30
compared with SIXBUILD tool 30
constraints 30
control file 30
description 30
sample audit log 32
segment requirements 30
table requirements 31
hrnbrtbl utility
and BATCHLOAD_CARDS tool 35
arguments 33
constraints 35
control file 34, 35
description 33
environment variable 35, 42, 49, 56
loading data into predefined tables 33
primary key 36, 117
sample audit log 39
secondary index 36

hrnbrula utility
arguments 41
constraints 42
control file 42
description 40
sample audit log 46
hrnbrulb utility
and BATCHUNLD_CARDS tool 49
arguments 48
compared with hrnbrulh utility 48
constraints 49
control file 49
description 48
-O option 123
sample audit log 52
hrnbrulh utility
and BATCHUNLD_CARDS tool 56
arguments 55
constraints 56
control file 56
description 54
-O option 123
sample audit log 60
hrncr file 105
hrncr subcommands
? 62
canceluser 62
checkpoint 62
dboffline 62
dbonline 62
kill 62
log 63
parm 63
resume 63
shutdown 63
spinsubmit 63
state 63
stop 63
stopserver 64
traceid 64
hrncr utility
arguments 61
description 61
environment variables 64
operating the Data Object Broker 61
subcommands 62
hrncrSvc utility
arguments 66
description 66
hrnspjex utility
arguments 68
description 68
environment variables 68
hrnspset utility
arguments 70
description 70
environment variables 71
hrntladm utility
arguments 72
description 72
environment variables 73
hrntlbps utility
arguments 74
description 74
environment variables 75
hrntlfcl utility
arguments 76
description 76
environment variables 77
hrntlfjr utility
arguments 79
description 79
environment variables 80
hrntlfps utility
arguments 82
description 82
environment variables 83
hrntlfrl utility
arguments 85
description 85
environment variables 86
hrntlmrg utility
arguments 87
description 87
environment variables 88

hrntlpcl utility
  description 89
hrntlrps utility
  arguments 90
  description 90
  environment variables 91
hrntrans utility *See* htrans utility
htrans utility
  arguments 93
  description 93
  examples 93
huron.dir 106

I

ID parameter, dbdef file 104
in-doubt transactions, printing 89
INIT parameter, dbdef file 104
input
  definition cards, defining 114
input to hrnbrptr utility 20, 20
Install Data Object Broker as a Windows Service utility
  description 66
Install osMon as a Windows Service utility
  description 98
installation, moving ACCESSLOG during 11

J

Journal Extraction utility
  arguments 68
  description 68
  environment variables 68
Journal Merge utility
  arguments 87
  description 87
  environment variables 88
Journal Spin utility
  arguments 68
  description 68
  environment variables 68

K

kill, hrncr subcommand 62

L

loading
  data into predefined tables 33
  TDS table 33
log, hrncr subcommand 63

M

Move ACCESSLOG utility
  arguments 12
  description 11
  prerequisites 11
  sample audit log 13
multiple secondary indexes, building in one pass 30

N

NAME parameter, dbdef file 103
null values 119

O

-O option 123
osBatch (Start an TIBCO Object Service Broker Batch Client) 94
osBatch utility, description 94
osMonSvc utility, description 98
output definition cards, defining 116

P

page fill tailoring guidelines 113
pages
  orphan, description 22
  referenced but indicated free 22
Pagestore Correction utility
  arguments 18
  description 18
  sample audit log 19
parameters, description of Data Object Broker parameter files 102
parm, hrncr subcommand 63
PATH parameter, dbdef file 104
primary key and hrnbrtbl utility 36, 117
Print Contingency Log utility, description 89
print utility, hrntlpcl 89
printing data from contingency log file 89

R

recovering table from archive 40
re-initialize segment utility 26
Reset Journal utility
  arguments 70
  description 70
  environment variables 71
Restore Pagestore utility
  arguments 90
  description 90
  environment variables 91
resume, hrncr subcommand 63
return codes
  hrnbrbal 6
  hrnbrnls 16
  hrnbrptr 24
rsview (Data Object Broker Resource Viewer) 99
rsview utility, description 99

S

secondary index
  and hrnbrtbl utility 36
  utility
    compared with SIXBUILD tool 30
    for TDS table 30
segment
  re-initialization utility 26
  requirements for
    hrnbrclr 9
    hrnbrptr 20
    hrnbrset 26
    hrnbrsix 30
Segment Balance utility
  arguments 4
  sample audit log 7
segment number
  dbdef file 104
shutdown, hrncr subcommand 63
SIXBUILD tool compared with hrnbrsix utility 30
SIXBUILD_CARDS tool, building control file for hrnbrsix utility 30
specification cards, defining 111
spinsubmit, hrncr subcommand 63
start an TIBCO Object Service Broker batch client utility, description 94
state, hrncr subcommand 63
stop, hrncr subcommand 63
stopserver, hrncr subcommand 64
support, contacting xvi
syntax specific null values 120

T

tables
  clearing with $CLRTAB tool 9
  clearing with hrnbrclr utility 9
  deleting selected instances using hrnbrclr utility 9
  instances, unloading 40
  loading data into predefined 33
  unloading from archive 40
tailoring page fill, guidelines 113

TDS
segment, clearing transient data from 26
table, loading 33
technical support xvi
TIBCO Object Service Broker directory 106
TIBCO_HOME xiii
tool, SIXBUILD compared with hrnbrsix utility 30
traceid, hrncr subcommand 64
transient data, clearing from a TDS segment 26
Translate Between Code Pages utility
arguments 14
constraints 15
control file 15
description 14
Translate Files utility
arguments 93
description 93
examples 93
translation between code pages 14
TYPE parameter, dbdef file 103

U

UNICODEDIR environment variable 35, 42, 49, 56
unload utilities, compared 54
unloading
table with segment offline 54
table with segment online 54
utilities
hrnbrclr (Batch Table Clear) 9
hrnbrial (Move ACCESSLOG) 11
hrnbrnls (Translate File Between Different Code Pages) 14
hrnbrpgc (Pagestore Correction) 18
hrnbrptr (Batch Pointer Check) 20
hrnbrset (Batch Segment Re-initialization) 26
hrnbrtbl (Batch Load) 33
hrnbrula (Batch Unload from Archive) 40
hrnbrulb (Batch Unload (offline)) 48
hrnbrulh (Batch Unload (online)) 54
hrncr (Data Object Broker) 61
hrncrSvc (Install Data Object Broker as a Windows Service) 66
hrnspjex (Journal Extraction (or Journal Spin)) 68
hrnspset (Reset Journal) 70
hrntladm (Administration Menu) 72
hrntlbps (Backup Pagestore) 74
hrntlfcl (Format Contingency Log) 76
hrntlfjr (Format Journal) 79
hrntlfps (Format Pagestore) 82
hrntlfrl (Format Redolog) 85
hrntlmrg (Journal Merge) 87
hrntlpcl (Print Contingency Log) 89
hrntlrps (Restore Pagestore) 90
hrntrans *See* htrans utility
htrans (Translate Files) 93
osMonSvc (Install osMon as a Windows Service) 98

V

value cards, defining 118
view resources and servers utility, description 99